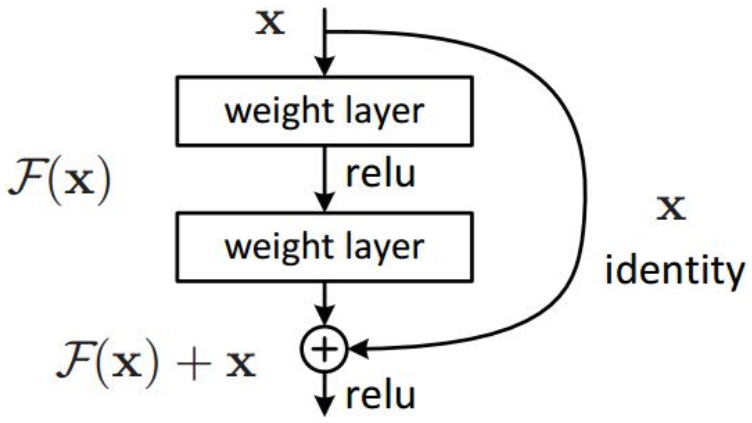
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

Clasificación y búsqueda por similitud de sketches usando Redes Convolucionales

CC6204: Deep Learning



Integrantes:
Nelson Higuera & Vicente Oyanadel
Profesores:
Jorge Pérez & José M. Saavedra
Fecha:
Domingo 29 de Junio, 2018.

Abstract

La extracción de características involucra reducir la cantidad de información necesaria para describir un dato; nos permite encontrar aquellas propiedades individuales (características) que describen mejor y compactamente los objetos analizados.

Estas son necesarias para entrenar clasificadores, y a veces incluso mejoran las interpretaciones humanas de cuáles son las características de un dato, sea el dominio que sea, las que aportan mayor información y permiten distinguir mejor entre distintos ejemplares.

Se entrenaron dos clasificadores basados en redes convolucionales, **skNet** y **skResNet** sobre un conjunto de dibujos hechos a mano. Luego las redes fueron utilizadas para extraer características, con las cuales fue calculado el *Mean Average Precision* (mAP) en el problema de Búsqueda por Similitud. **skResNet** se diferencia de la primera por el uso de capas residuales en su arquitectura.

La **accuracy** obtenida para el problema de clasificación en 100 clases, para los datos de entrenamiento y prueba fueron **skNet** de 0.844 y 0.5054, y de **skResNet** de 0.896 y 0.496, respectivamente. El **mAP** calculado para la **skNet** fue de 0.202 y para **skResNet** fue de 0.200.

El código fuente está disponible en GitHub: https://github.com/Vichoko/ConvSketch
Los TFRecords y modelos entrenados se encuentran en:
https://users.dcc.uchile.cl/~voyanede/cc6204/quickdraw/

Introducción

Hasta antes de la invención de las redes profundas, la ingeniería de características era un trabajo que requería de recursos humanos especializados en el dominio en cuestión. Sin embargo, las redes profundas han demostrado, además de ser clasificadores potentes, servir de extractores de características, o *Deep Features,* de calidad y sin necesidad de intervención humana.

En la presente tarea se utilizan dos arquitecturas de redes convolucionales que clasifican dibujos a mano alzada (Sketches), dentro 100 clases distintas. Comparamos el desempeño de ambas arquitecturas en la clasificación de estas imágenes mediante métricas de **Accuracy** y **Loss**.

Se siguen los siguientes pasos:

1. La construcción de dos conjuntos de datos, para entrenar (*train-set*) y evaluar (*test-set*), tomando como base el dataset público de dibujos a mano alzada o Sketches “The Quick, Draw!” de Google
2. El entrenamiento de dos clasificadores basados en redes convolucionales **skNet** y **skResNet**, y la comparación de sus métricas de desempeño.
3. La utilización de las redes previamente entrenadas para extraer características, y luego evaluar la calidad de estas características en el problema de Búsqueda por Similitud mediante mAP.

Para evaluar el **mAP**, se extraen las características del conjunto de datos de prueba (~5000 vectores de largo 1024), mediante el modelo entrenado previamente, y se usa la estrategia *leave one out* para evaluar la efectividad para recuperar el resto de las imágenes de su misma clase. La función de distancia usada es norma L2.

Se espera poder obtener resultados que muestran cómo el uso de redes convolucionales permite la extracción de características importantes en los objetos estudiados.

Además, se espera un mejor desempeño general de la red convolucional residual, en comparación con la red convolucional tradicional.

Desarrollo

Construcción del conjunto de datos

Se obtienen los datos desde el repositorio de pinceladas de dibujos a mano alzada obtenido y publicado por Google (*The Quick, Draw! Dataset*). Este consta de 50 millones de dibujos repartidos en 345 categorías. Sin embargo, en esta tarea utilizaremos solamente 100 categorías, que serán elegidas aleatoriamente.

Los archivos de datos están estructurados como trazos o pinceladas sobre un lienzo, en un formato serializado (*.ndjson*). Los cuales deben ser transformados a un mapa de bits para ser compatibles con la arquitectura convolucional. Para ello, se programó una función la cual transforma los archivos obtenidos del repositorio a mapas de bits, los cuales son convolucionables.

En primer lugar se seleccionan 100 clases del conjunto total de clases disponibles, luego se transforman los archivos a un formato de mapa de bits, y finalmente, se serializan a TFRecords, un formato fácilmente importable por TensorFlow.

Estos TFRecords pueden descargarse aquí:
https://users.dcc.uchile.cl/~voyanede/cc6204/quickdraw/data/

El código fuente de la carga, transformación y serialización de los archivos del dataset se encuentra en la clase: *images-load-transform-serializer.py*

Clasificación de Sketches

Se construyeron dos clasificadores basados en redes convolucionales, **skNet** y **skResNet**. La diferencia importante entre ambas, es que **skResNet** ocupa capas residuales que consisten en la acumulacion de los resultados de las capas anteriores. Para una definición en detalle de la arquitectura de ambas redes, diríjase al enunciado. Ambas se entrenaron ocupando el conjunto de datos de entrenamiento (*train-set)*, utilizando los hiperparametros siguientes:

- Tasa de aprendizaje: 0.0001
- Número de iteraciones: 10000
- Tamaño de los batches: 10

Mientras se entrena sobre los datos de entrenamiento se muestran métricas obtenidas sobre el mismo conjunto de datos de entrenamiento, cómo **Accuracy** y **Loss**.

Una vez entrenados se evalúan los modelos mediante la predicción sobre el conjunto de datos de entrenamiento (*train-set)* y de evaluación (*test-set),* obteniendo la Accuracy y el Loss sobre ambos conjuntos

Los modelos entrenados se almacenan en disco para el próximo paso.

Los modelos entrenados se pueden descargar aquí:
https://users.dcc.uchile.cl/~voyanede/cc6204/quickdraw/models/sknet/
https://users.dcc.uchile.cl/~voyanede/cc6204/quickdraw/models/skresnet/

El código fuente del entrenamiento y evaluación de los modelos se encuentra en:
train_sketch_net.py

Para entrenar el modelo con los datos de entrenamiento, hay que ejecutar esta clase con el parámetro: *-mode train*

Para evaluar el modelo con los datos de prueba, hay que ejecutar esta clase con el parámetro: *-mode evaluate*

Para el correcto funcionamiento hay que configurar la ruta de los *TFRecords* en el archivo: *configuration_sketch.py.* Particularmente la variable *DATA_PATH*

Búsqueda por similitud

El conjunto de datos de evaluación (*test-set)* es utilizado para extraer sus características.

El proceso consiste en tomar una red y hacer una predicción utilizando como entrada un elemento del conjunto de datos. Se obtienen los valores de salida de la penúltima capa *fully-connected*, los cuales corresponden a un arreglo de 1024 valores. Estas son las características extraídas del input en cuestión; obteniendo finalmente 50 vectores por cada clase.

Obtenemos esas características para cada clase del conjunto, es decir, tenemos 5000 vectores de características (50x100 clases).

Luego, para cada vector de características del conjunto de datos de prueba, se evalúa el desempeño para recuperar el resto de los ejemplos de la misma clase sobre el conjunto de vectores; obteniendo y guardando el Average Precision para cada uno.

Finalmente, al obtener el Average Precision de cada vector, se promedian y se obtienen el mAP. Esta métrica logra evaluar el desempeño total del sistema de búsqueda por similitud, en la tarea de recobrar ejemplares de todos los tipos de clases existentes en el conjunto de datos de prueba. Así obtenemos una medida del mAP de la **skNet** y **skRestNet**.

El código fuente de la extracción de características se encuentra en el archivo: *train_sketch_net.py* (linea 124)

El codigo fuente de el calculo de la metrica mAP se encuentra en el archivo: *evaluate_similarity_lookup.py*

Para extraer las características del conjunto de prueba y obtener el mAP, hay que ejecutar la clase *train_sketch_net.py* con el parámetro *-mode test.*

Resultados Experimentales y Discusión

Resultados

Los resultados del entrenamiento para cada una de las redes fueron los siguientes.

Accuracy y Loss en datos de entrenamiento:

- skNet: 0.844; 0.463
- skResNet: 0.896; 0.320

Accuracy y Loss en datos de prueba:

- skNet: 0.5054; 2.129
- skResNet: 0.496; 2.308

El mAP obtenido sobre datos de prueba:

- mAP skNet: 0.202
- mAP skResNet: 0.200

Discusión

Considerando que eran 100 clases, el accuracy alcanzado sobre el conjunto de datos de prueba, implica que el modelo predice la clase correcta aproximadamente la mitad de las veces. Esto lo consideramos aceptable, teniendo en cuenta la gran cantidad de clases que podría ser.

Notamos que ambas redes tuvieron rendimientos muy similares, a pesar de las diferencias en su arquitectura. Esto se puede deber a que las optimizaciones de la ResNet, representan una optimización para el entrenamiento de redes profundas (+152 capas).

Por otro lado, vemos que el mAP exhibe el mismo esquema anterior, las distintas arquitecturas no introducen gran diferencia en la métrica. Esto no debería sorprendernos ya que si ambas tenían un accuracy similar, se espera que el mAP también tenga valores similares. Esto dado que las características que extrajimos de cada red son ocupadas por las mismas redes en la predicción, lo que significa que la calidad de las características afectan directamente el accuracy.

Conclusiones

La experiencia permitió familiarizarnos con la librería TensorFlow; con la construcción y entrenamiento de redes convolucionales; y la extracción de características usando las redes entrenadas.

Estamos satisfechos con el trabajo desarrollado, dado que se logró implementar exitosamente la carga y transformación de datos; el entrenamiento y evaluación de ambas redes; la extracción de características de los modelos entrenados; y la evaluación del sistema de Búsqueda por Similitud.

Con respecto a los resultados de la clasificación, se concluye que la precisión alcanzada (~50%) es muy aceptable, considerando la gran cantidad de clases posibles.

Por otro lado, relativo a los resultados obtenidos de la Búsqueda por Similitud, consideramos que el mAP alcanzado (~20%) es bajo para lo que uno espera de un sistema de búsqueda realmente útil. Sin embargo, el valor alcanzado nos parece consecuente con la calidad del clasificador.

Respectivo a las arquitecturas utilizadas, se esperó un mejor rendimiento para la arquitectura residual, dados los buenos resultados obtenidos con el dataset de ImageNet (Kaiming He, et Al.), pero sorprendentemente los resultados fueron muy similares a una arquitectura convolucional estándar.

Se concluye que la causa probable de esta similaridad, es que las optimizaciones introducidas por la Red Residual en su publicación original surgen como una solución para el entrenamiento de redes profundas (Notemos que la resNet publicada cuenta con 152 capas). Sabiendo esto, consideramos que nuestras redes pueden ser muy superficiales para notar las optimizaciones introducida por esta arquitectura.