Presentación final: Machine Learning

Nombre: José Estuardo Menéndez Carnet: 18072

Junio 2021

Introducción

Una de las acciones modernas que más tración han generado es la iniciativa para la aceptación de las personas sin importar las discapacidades que este tenga. Es importante mostrarle a la gente que no importa si hay ciertas diferencias en capacidades, es posible superarse y lograr los objetivos que se pongan. Uno de estos grupos es la gente sorda, la cual es de las más afectadas por su discapacidad. Actualmente un 5% de las personas en el mundo son sordas, y de ese 5%, un 27% viven en paises de tercer mundo como Guatemala.

El problema con esto es que no hay suficiente consciencia social sobre estra problemática, por lo que muchas veces es ignorada. Existen fundaciones y comites que tiene como objetivo ayudar a las personas sordas para poder aprender lenguaje de señas y poder comunicarse de manera fácil con los demás, pero el problema es que no mucha gente sabe hablar lenguaje de señas, en especial considerando que cada pais tiene diferencias significativas entre lenguajes, lo cual imposibilita la posibilidad de exportar instructores para impartir clases.

Objetivos

- Identificar cual método de clasificación de imágenes es más efectivo para la elaboración de clasificación de imagenes de lenguaje de señas.
- Evaluar la posibilidad de llevar el proyecto a mayor escala para crear un traductor funcional.

Procedimiento experimental

En esta ocación se hizo uso de la base de datos "Hand Gesture Recognition Database", la cual, como su nombre lo indica; es una base de datos de gestos de manos para evaluar algoritmos de reconocimiento de imagenes. Dado que se haría un proceso de clasificación, no se hizo mayor cambio a la base de datos para utilizarla. Los únicos cambios que se hizieron fue cambio de resolución de la imagen, conviertiendola en imagen de 150×150 y pasarle algunos filtros para que los algoritmos tuvieran mayor facilidad para procesar la imagen. Con la intención de evaluar el desempeño de una red neuronal convolucional para la clasificación de este tipo de imágenes se tomará como marco de referencia el algoritmo de Random Forest Classifier, el cual también es sumamente efectivo a la hora de clasificar imágenes.

Metodología

Como se mencionó, se hizo uso de 2 algoritmos de clasificación cuando se evaluó, una red neuronal convolucional (CNN) y un clasificador Random Forest. La arquitectura para el CNN fue hacer uso de 4 capas ocultas. Las capas se explican con mayor énfasis a continuación:

- Capa 1: La primera capa tiene 32 filtros, un kernel de tamaño 5×5 y una función de activación relu. Además se agregó un max pooling con pool size de 2×2.

- Capa 2: La siguiente capa tiene 64 filtros, un kernel de tamaño 3×3, función de activación relu y un max pooling con polling size de 2×2 y Strides de 2×2. Todos los demás max pooling fueron los mismos, por lo que ya no menciona en las otras capas.
- Capas 3 y 4: Estas tienen 96 filtros, un kernel de tamaño 3×3 y función de activación relu.

Finalmente se hizo un Flatten para aplanar los datos, una capa de Dense para conectar todos los nodos, una capa de Activation para usar una activación relu y por último se pasó otra capa de Dense con activación softmax.

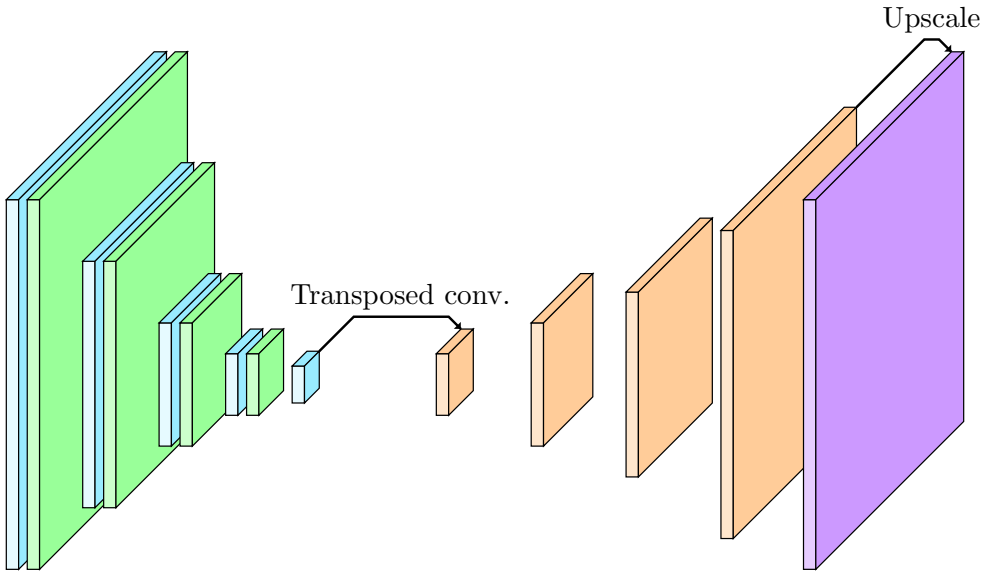

Figure 1: Red neuronal creada. Las capas celestes representan los Conv2D, mientras que las verdes son las capas de Max Pooling. Las capas naranja son las capas finales, siendo la última capa naranja la que tiene el upscale y el softmax.

Por otra parte, el Random Forest fue un algoritmo sumamente sencillo en terminos generales. Se usaron los valores predeterminnados que SKlearning daba, los cuales obtuvieron los mejores resultados. Se hicieron 100 ramas, sin una profundidad máxima, split mínimo de 2 hojas y hoja mínima de 1.

Resultados

```
Epoch 1/10
118/118 [==============================] - 442s 4s/step - loss: 0.9198 - accuracy: 0.6897 - val_loss: 0.0115 - val_accuracy: 0.9982

Epoch 00001: val_loss improved from inf to 0.01153, saving model to ./base.model
Epoch 2/10
118/118 [==============================] - 426s 4s/step - loss: 0.0061 - accuracy: 0.9983 - val_loss: 0.0114 - val_accuracy: 0.9982

Epoch 00002: val_loss improved from 0.01153 to 0.01136, saving model to ./base.model
Epoch 3/10
118/118 [==============================] - 442s 4s/step - loss: 0.0047 - accuracy: 0.9993 - val_loss: 0.1619 - val_accuracy: 0.9496

Epoch 00003: val_loss did not improve from 0.01136
Epoch 4/10
118/118 [==============================] - 433s 4s/step - loss: 0.0379 - accuracy: 0.9879 - val_loss: 0.0042 - val_accuracy: 0.9992

Epoch 00004: val_loss improved from 0.01136 to 0.00420, saving model to ./base.model
Epoch 5/10
118/118 [==============================] - 423s 4s/step - loss: 2.1534e-04 - accuracy: 1.0000 - val_loss: 0.0026 - val_accuracy: 0.9994

Epoch 00005: val_loss improved from 0.00420 to 0.00259, saving model to ./base.model
Epoch 6/10
118/118 [==============================] - 433s 4s/step - loss: 1.9527e-04 - accuracy: 1.0000 - val_loss: 0.0027 - val_accuracy: 0.9996

Epoch 00006: val_loss did not improve from 0.00259
Epoch 7/10
118/118 [==============================] - 436s 4s/step - loss: 1.3191e-05 - accuracy: 1.0000 - val_loss: 0.0025 - val_accuracy: 0.9996

Epoch 00007: val_loss improved from 0.00259 to 0.00255, saving model to ./base.model
Epoch 8/10
118/118 [==============================] - 433s 4s/step - loss: 1.0113e-05 - accuracy: 1.0000 - val_loss: 0.0025 - val_accuracy: 0.9996

Epoch 00008: val_loss improved from 0.00255 to 0.00247, saving model to ./base.model
Epoch 9/10
118/118 [==============================] - 423s 4s/step - loss: 6.0796e-06 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 0.9996

Epoch 00009: val_loss improved from 0.00247 to 0.00241, saving model to ./base.model
Epoch 10/10
118/118 [==============================] - 437s 4s/step - loss: 5.0922e-06 - accuracy: 1.0000 - val_loss: 0.0024 - val_accuracy: 0.9996

Epoch 00010: val_loss improved from 0.00241 to 0.00236, saving model to ./base.model
```

Figure 2: Generaciones de la red neuronal

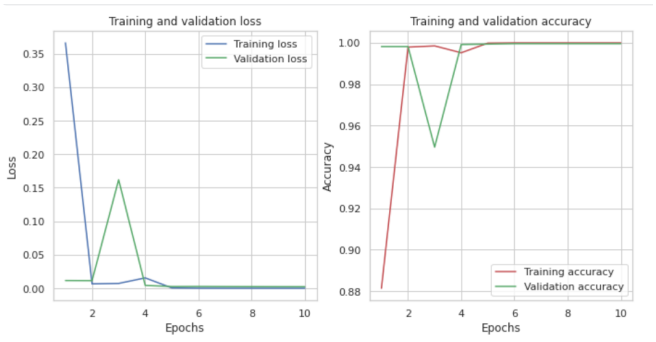

Figure 3: Eficacia del CNN durante las generaciones

Training metrics:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1482
1	1.00	1.00	1.00	1489
2	1.00	1.00	1.00	1497
3	1.00	1.00	1.00	1492
4	1.00	1.00	1.00	1534
5	1.00	1.00	1.00	1510
6	1.00	1.00	1.00	1501
7	1.00	1.00	1.00	1498
8	1.00	1.00	1.00	1482
9	1.00	1.00	1.00	1515
micro avg	1.00	1.00	1.00	15000
macro avg	1.00	1.00	1.00	15000
weighted avg	1.00	1.00	1.00	15000
samples avg	1.00	1.00	1.00	15000

Figure 4: Métricas de los datos de entrenamiento con RF

Test data metrics:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	518
1	1.00	0.99	1.00	511
2	1.00	1.00	1.00	503
3	1.00	1.00	1.00	508
4	1.00	1.00	1.00	466
5	1.00	1.00	1.00	490
6	1.00	1.00	1.00	499
7	1.00	1.00	1.00	502
8	1.00	0.99	1.00	518
9	1.00	1.00	1.00	485
micro avg	1.00	1.00	1.00	5000
macro avg	1.00	1.00	1.00	5000
weighted avg	1.00	1.00	1.00	5000
samples avg	1.00	1.00	1.00	5000

Figure 5: Métricas de los datos de testing con RF

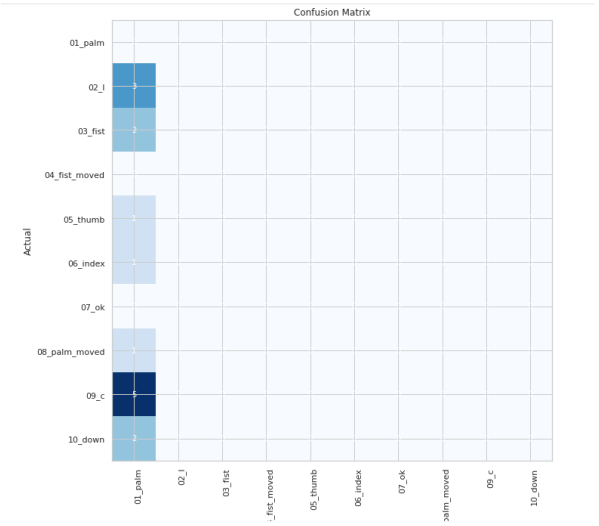

Figure 6: Matriz de confusión de RF

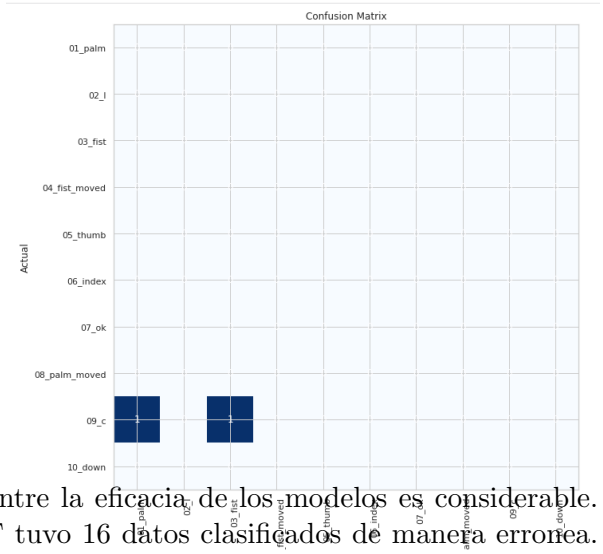

Figure 7: Matriz de confusión de CNN

Discusión

Como se puede observar en la figura 6 y 7 la diferencia entre la eficacia de los modelos es considerable. Mientras el CNN solo tuvo 2 datos mal clasificados, el RF tuvo 16 datos clasificados de manera errónea. Este hecho, sin embargo, no es indicativo que el modelo de RF es malo ya que, como se puede ver en la figura 5, las métricas de e proceso de testing es casi perfecto en todos los datos. Al observa de cerca los datos, notamos que 16 de 5000 datos clasificados de manera errónea no es un mal modelo. Ambos modelos fueron sumamente efectivos a la hora de clasificar en sus respectivos testings.

Por otra parte, en terminos de simpleza del algoritmo y tiempo de compilamiento , el Random forest fue superior al CNN dado que su tiempo de compilamiento fue de 10 minutos, mientras que el de CNN tardó al rededor de 90 minutos. En caso de querer utilizar este modelo en una escala más grande, para una base de datos con más objetos que clasificar, es mejor considerar utilizar el modelo de Random Forest o mejorar la arquitectura del modelo de CNN para que este no tarde tanto.

Conclusiones

- Se demostró que en términos de accuracy, el modelo de CNN es más efectivo a la hora de clasificar las imágenes
- Se observa que en términos de rapidez, el modelo de random forest es significativamente más rápido que el modelo de CNN
- Se encontró una cantidad de datos apropiados para hacer uso de un algoritmo de clasificación de imágenes para entrenamiento de un modelo.

Referencias

Link para Base de datos: https://www.kaggle.com/gti-upm/leapgestrecog