Sistema de reconocimiento e identificacion facial aplicado a la seguridad laboral

Edgar Montenegro,
Leyston Onate &
Franklin Sierra

Motion Analysis and Computer Vision
Universidad Industrial de Santander
Bucaramanga - Colombia

Universidad Industrial de Santander

MACV
Motion Analysis and Computer Vision

BIVL²ab
Biomedical Imaging, Vision and Learning Laboratory

Tabla de contenido

Motivaciones y Objetivos

Motivation

1. Actualmente existen entornos carentes de herramientas tecnologicas que puedan brindar un soporte de ingreso e egreso a usuarios de alto nivel.

Fig 1. Seguridad digital

Motivation

2. Ofrecer un seguimiento robusto en recintos donde la permanencia es obligatoria en ciertos periodos de tiempo.
3. Presentar un modelo operacional que muestre la utilidad de este modelo de seguridad.

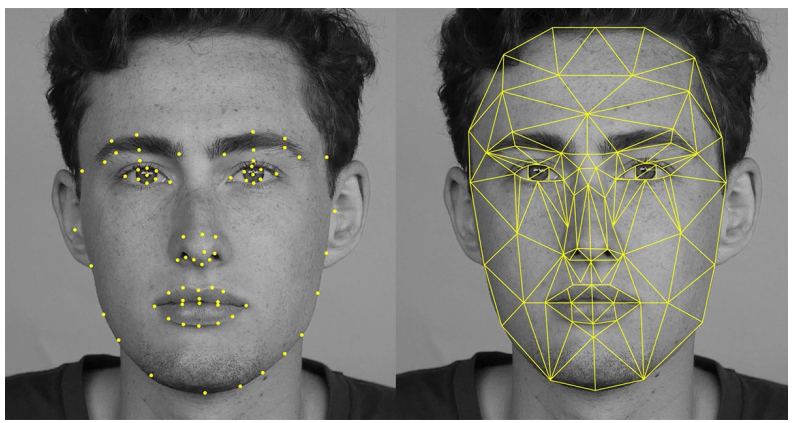


Fig 2. Reconocimiento facial

Objetivos

- Crear un dispositivo de seguridad que integre conocimientos de microcontroladores y visión por computadora.
- Implementar diversas tecnicas de detección y clasificación facial con OpenCV y redes neuronales.
- Llevar el prototipo a un entorno real donde se pueda evidenciar su confiabilidad por medio de la implementacin física.

Modelo

- Sistema de seguridad con integración de los estudios de visión por computadora, específicamente, el reconocimiento y identificación de rostros con el uso de deep learning.

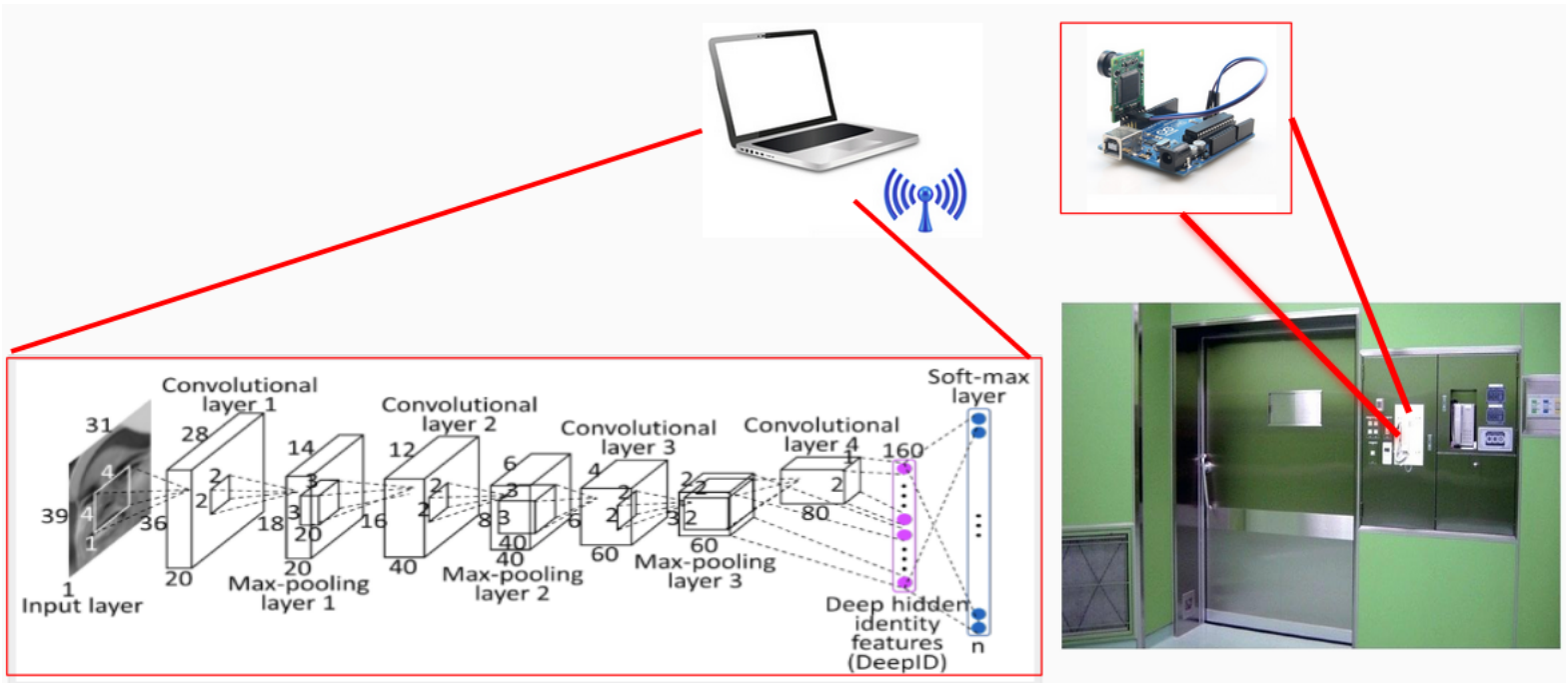
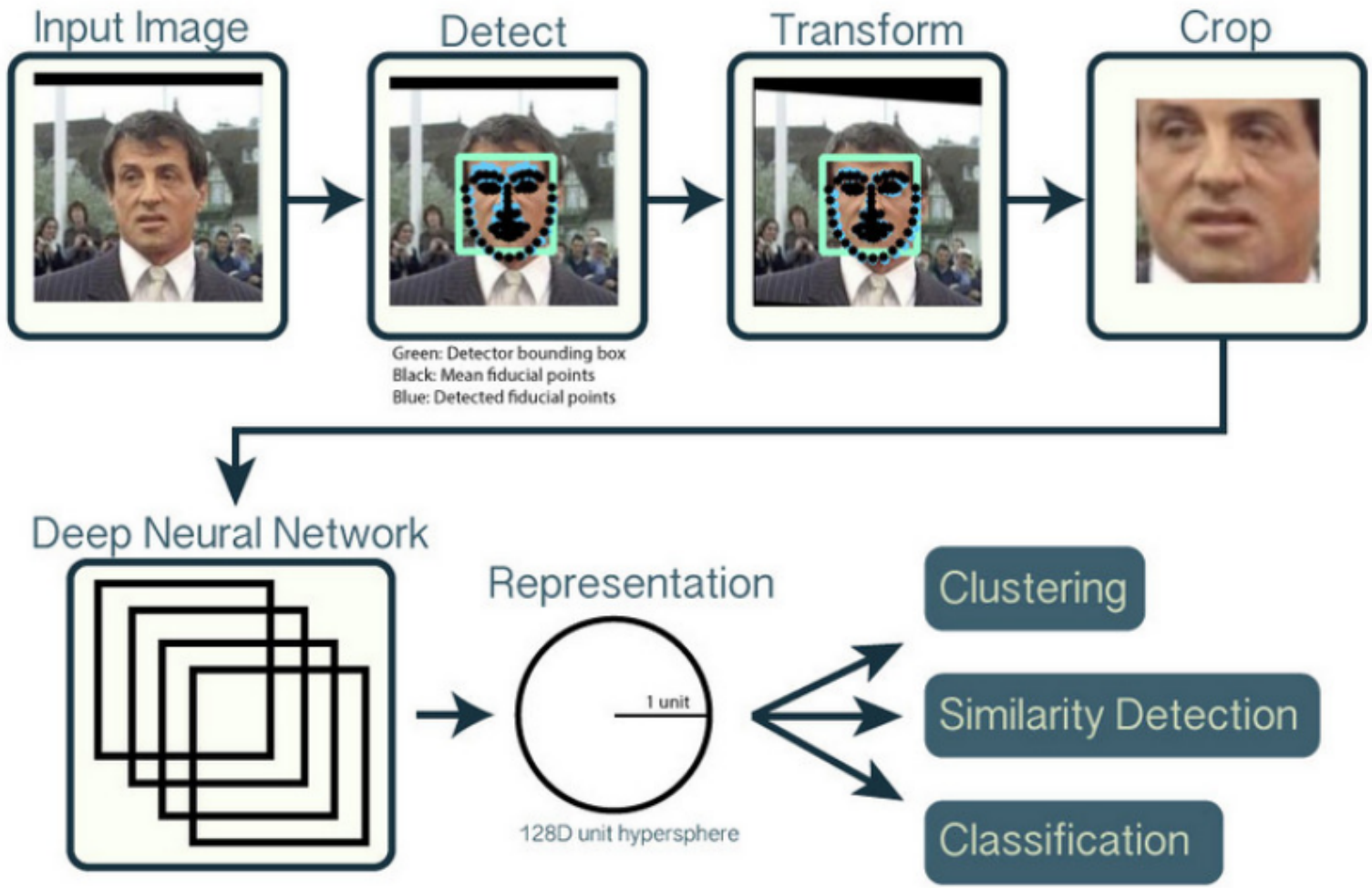


Fig 3. Modelo de aplicacion

Metodologia

Reconocimento facial



Keypoints y descriptores asociados

- Son las localizaciones especiales, o puntos en la imagenes que definen que es interesante o que resalta en la imagen.

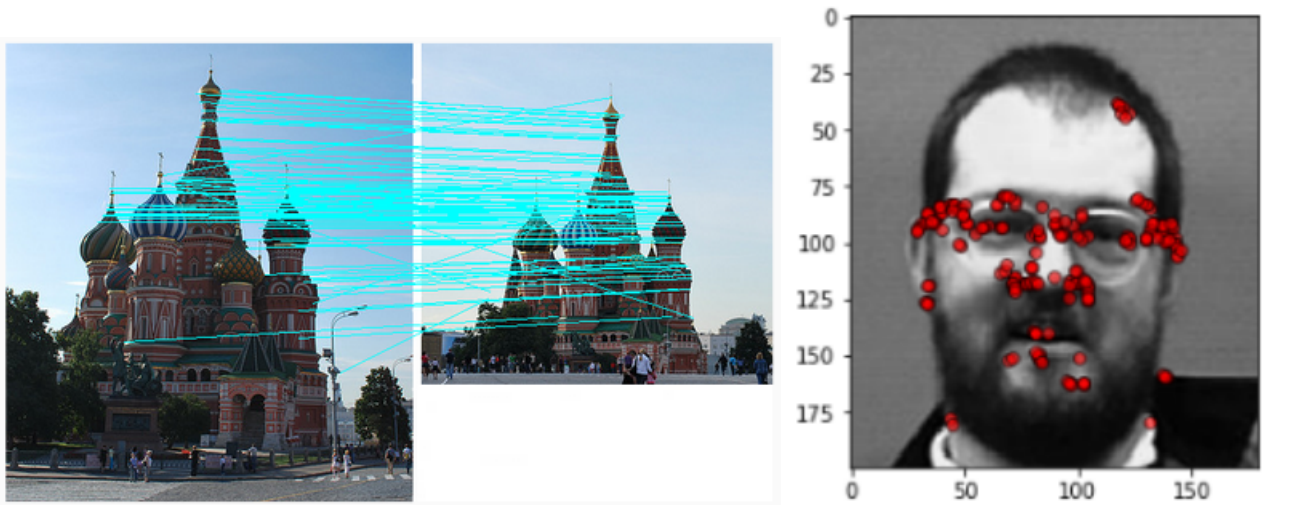


Fig 4. Keypoints

Convolutional neuronal networks - CNN

- Un modelo de redes neuronales convolucionales se utilizara para el análisis de la información de las imagenes.

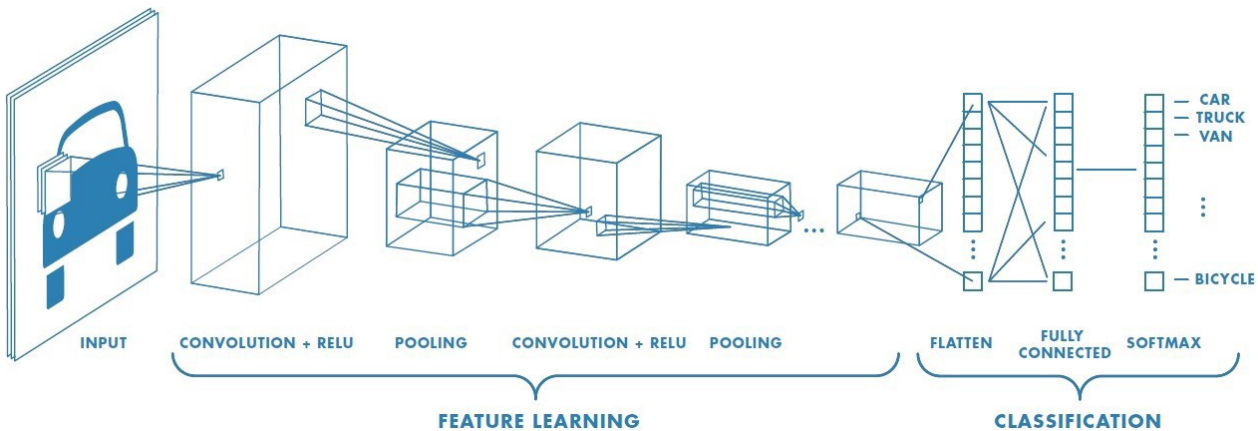


Fig 5. Modelo de CNN

Dataset

Viejo Dataset

Las imagenes de este dataset son muy parecidas entre si, generando problemas de sobre entrenamiento:

- 3040 imagenes.
- 152 personas.
- 20 imagenes por persona.

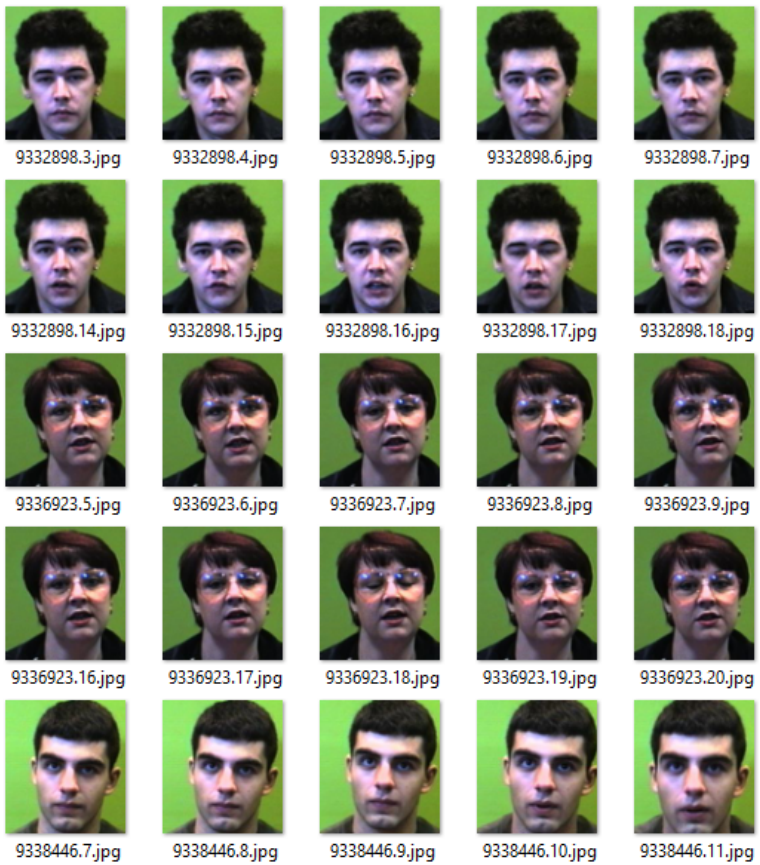


Fig 6. Dataset

Nuevo Dataset

Para el análisis de los datos se utilizó un dataset con:

- 4000 imagenes, de 1000 personas, 4 imagenes por persona.
- En los experimentos se usaron 40 personas.

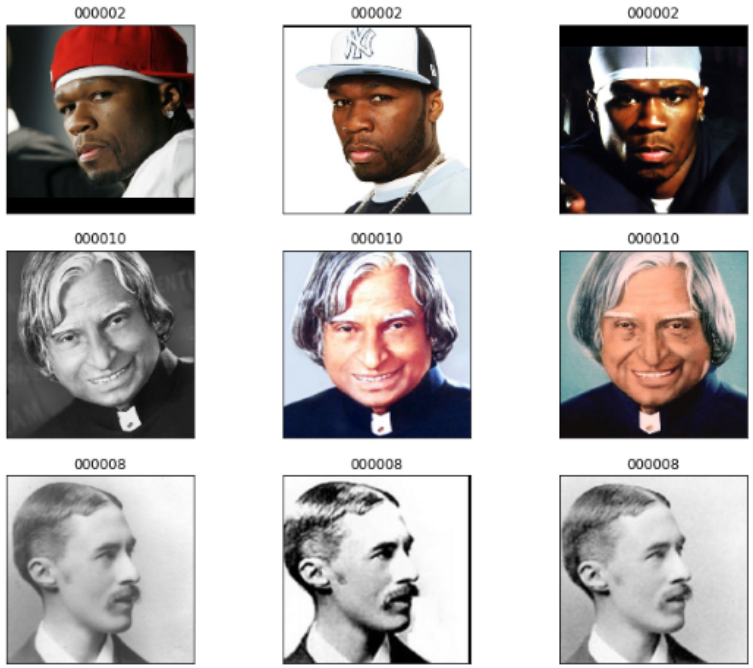


Fig 6. New Dataset

Resultados

Problemas encontrados

- Durante nuestros estudios iniciales experimentamos problematicas debido a la impresiza separacion de los datos .

```
[ ]  X_mix = keras.applications.mobilenet.preprocess_input(X_mix)

     test_size = 0.2
     X_train, X_test, y_train, y_test = train_test_split(X_mix, Y_mix, test_size=0.2, random_state=42)
     print  X_test.shape,  X_train.shape
```

Resultados erroneos

- Resultados de la clasifcación de las imagenes del dataset viejo

Metodo	Accuracy	Validadcion accuracy
KNN	0.9539	——
CNN sin filtros	0.0074	0.0000
CNN con filtros	1.0000	0.9843

Nuevo modelo de analisis

- Para satisfacer las necesidades de nuestro modelo de analisis fue necesario necesario modificar como se obtenian los datos

```
[6]  def div(pers, labels_clas):
       datos_train = []
       datos_x_test =[]
       datos_y_test =[]
       datos_x_train =[]
       datos_y_train =[]

       for i in range(0,len(pers),4):
         x_test = []
         y_test = []
         x_train = []
         y_train = []
         x_traint_total = []
         y_traint_total = []

         for j in range(i,i+2):
           x_test.append(pers[j])
           y_test.append(labels_clas[j])
         for j in range(i+2,i+4):
           x_train.append(pers[j])
           y_train.append(labels_clas[j])
         x_traint_total.extend(pers[:i])
         x_traint_total.extend(x_train)
         x_traint_total.extend(pers[i+4:])
         y_traint_total.extend(labels_clas[:i])
         y_traint_total.extend(y_train)
```

Transfer Learning

- Metodologia de deep learning en donde se utiliza modelos pre entrenados para realizar analisis de deatasets.

CNN-based FR

VGG Face (2015)

FaceNet (Google 2015)

Data Augmentation

- Metodologia de deep learning para crear mas datos apartir de datos existentes con la ayuda de los metodos de trasformacion de una imagen.

```
[13] datagen = ImageDataGenerator()
     datagen.fit(datos_x_train)

     datagen2 = ImageDataGenerator()
     datagen2.fit(datos_x_test)

     it_train = datagen.flow(datos_x_train, datos_y_train, batch_size=100)
     it_test = datagen2.flow(datos_x_test, datos_y_test, batch_size=20)
```

Resultados Nuevos

- Resultados de la clasifcación de las imagenes del dataset Nuevo con las nuevas implementaciones.

Metodo	Accuracy	Validadcion accuracy
CNN propia	0.9999	1.0000
CNN MovilNet	1.0000	0.6000

Clasificador CNN Sequential

- Las imagenes se pasaron al siguiente modelo de clasificacion

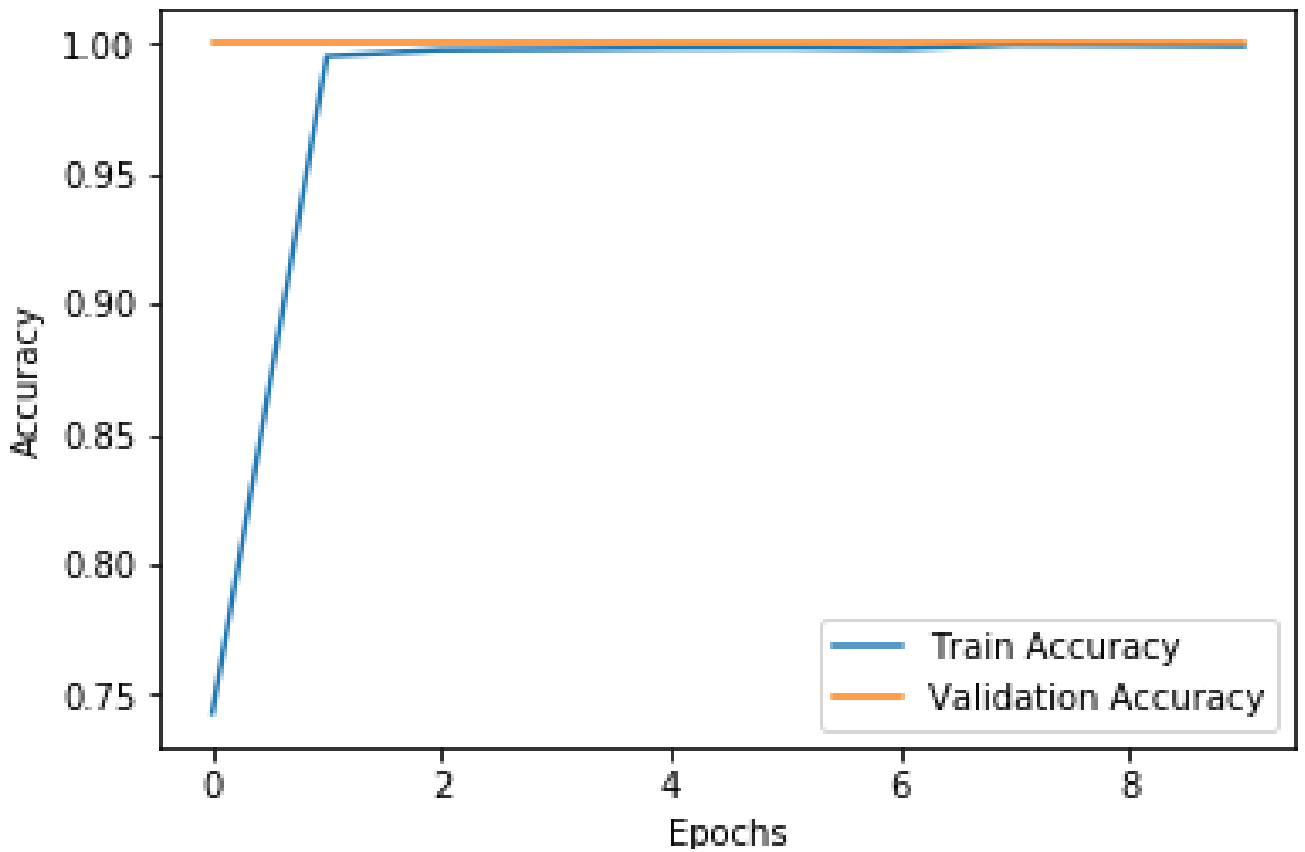
```
[17] opt = tf.keras.optimizers.SGD(lr=0.001, momentum=0.9)
     model_dr.compile(optimizer=opt, loss='sparse_categorical_crossentropy',metrics=['accuracy'])

     history = model_dr.fit_generator(it_train, steps_per_epoch=100, epochs=10, verbose=1, validation_data=it_test,validation_steps=20,
```

```
Epoch 1/10
100/100 [==============================] - 86s 857ms/step - loss: 1.0761 - acc: 0.7429 - val_loss: 0.0205 - val_acc: 1.0000
Epoch 2/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0170 - acc: 0.9951 - val_loss: 0.0015 - val_acc: 1.0000
Epoch 3/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0084 - acc: 0.9975 - val_loss: 0.0021 - val_acc: 1.0000
Epoch 4/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0075 - acc: 0.9978 - val_loss: 8.6222e-04 - val_acc: 1.0000
Epoch 5/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0064 - acc: 0.9982 - val_loss: 5.6209e-04 - val_acc: 1.0000
Epoch 6/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0045 - acc: 0.9983 - val_loss: 2.9768e-05 - val_acc: 1.0000
Epoch 7/10
100/100 [==============================] - 74s 741ms/step - loss: 0.0063 - acc: 0.9979 - val_loss: 0.0065 - val_acc: 1.0000
Epoch 8/10
100/100 [==============================] - 74s 742ms/step - loss: 0.0021 - acc: 0.9995 - val_loss: 1.0865e-04 - val_acc: 1.0000
Epoch 9/10
100/100 [==============================] - 74s 742ms/step - loss: 0.0023 - acc: 0.9993 - val_loss: 2.7178e-04 - val_acc: 1.0000
```

Clasificador CNN Sequential

- Representación grafica.

```
<matplotlib.legend.Legend at 0x7fd460a6ca50>
```



Clasificador CNN MobilNet

- Las imagenes se pasaron al siguiente modelo de clasificacion utilizando trasfer learning

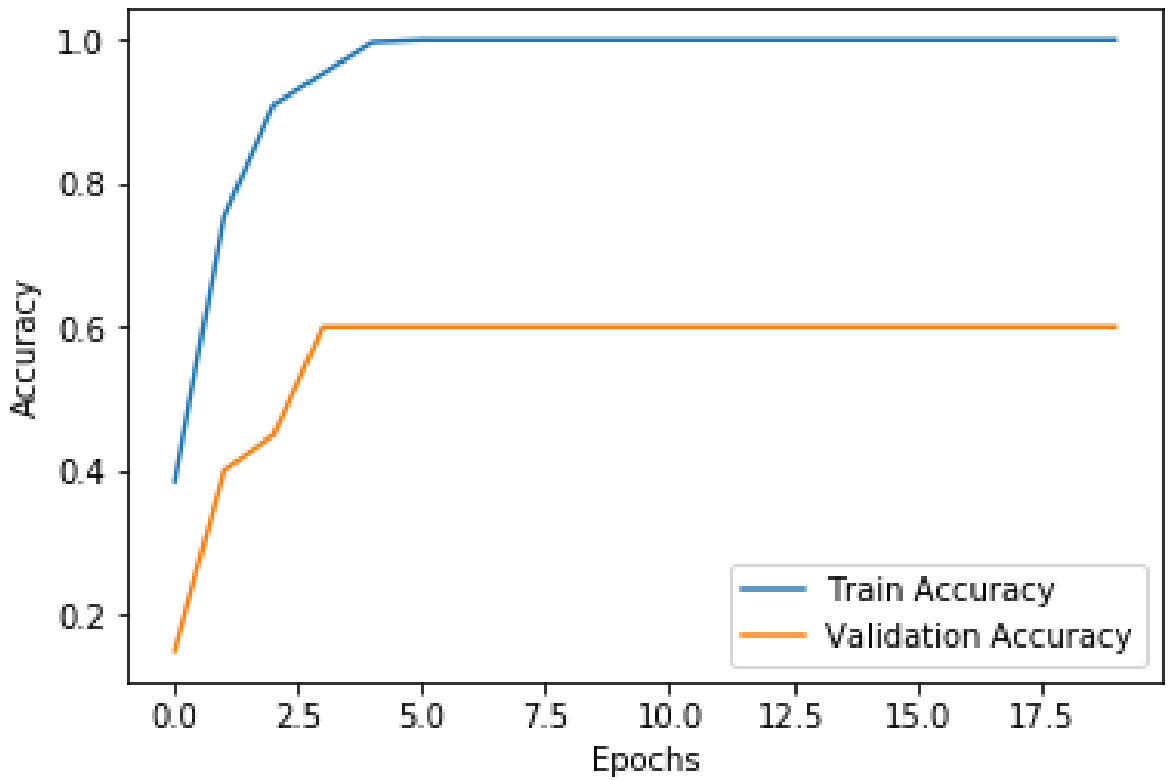
```
[14] opt = tf.keras.optimizers.SGD(lr=0.001, momentum=0.9)
     model_B_on_A.compile(optimizer=opt, loss='sparse_categorical_crossentropy',metrics=['accuracy'])

     history = model_B_on_A.fit_generator(it_train, steps_per_epoch=100, epochs=20, verbose=1, validation_data=it_test,validation_steps=20,

     #history = model_B_on_A.fit(datos_x_train, datos_y_train, epochs=50, batch_size=32,validation_data=(datos_x_test, datos_y_test))
```

```
W0821 20:51:43.242592 140553177532288 deprecation.py:323] From /usr/local/lib/python2.7/dist-packages/tensorflow/python/ops/math_grad.p
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Epoch 1/20
100/100 [==============================] - 27s 274ms/step - loss: 2.6811 - acc: 0.3858 - val_loss: 3.3921 - val_acc: 0.1500
Epoch 2/20
100/100 [==============================] - 23s 233ms/step - loss: 1.3684 - acc: 0.7536 - val_loss: 2.7963 - val_acc: 0.4000
Epoch 3/20
100/100 [==============================] - 23s 231ms/step - loss: 0.5668 - acc: 0.9094 - val_loss: 2.6560 - val_acc: 0.4500
Epoch 4/20
100/100 [==============================] - 23s 231ms/step - loss: 0.2980 - acc: 0.9527 - val_loss: 2.2642 - val_acc: 0.6000
Epoch 5/20
100/100 [==============================] - 23s 231ms/step - loss: 0.1147 - acc: 0.9965 - val_loss: 2.2040 - val_acc: 0.6000
Epoch 6/20
100/100 [==============================] - 23s 232ms/step - loss: 0.0369 - acc: 1.0000 - val_loss: 2.2240 - val_acc: 0.6000
Epoch 7/20
100/100 [==============================] - 23s 232ms/step - loss: 0.0253 - acc: 1.0000 - val_loss: 2.1592 - val_acc: 0.6000
Epoch 8/20
100/100 [==============================] - 23s 232ms/step - loss: 0.0198 - acc: 1.0000 - val_loss: 2.1162 - val_acc: 0.6000
Epoch 9/20
100/100 [==============================] - 23s 231ms/step - loss: 0.0164 - acc: 1.0000 - val_loss: 2.0703 - val_acc: 0.6000
```

Clasificador CNN MobilNet

- Representación grafica.

```
<matplotlib.legend.Legend at 0x7fd45d6adc90>
```



Thank you for your attention!