

Identificación del número de fachadas en las viviendas

- Manuel Jesús Casillas Olivier

**Temas Selectos de Física
Computacional III
Semestre 2024-1
11/12/2023**

Obtención de Datos SVHN

Los datos se obtuvieron de la siguiente base de TensorFlow:

TensorFlow > Recursos > Datasets > Catalog

¿Te resultó útil?  

svhn_recortado 

- Descripción :

El conjunto de datos de Street View House Numbers (SVHN) es un conjunto de datos de reconocimiento de dígitos de imágenes de más de 600 000 imágenes de dígitos provenientes de datos del mundo real. Las imágenes se recortan a 32x32.

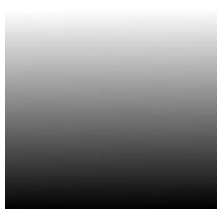
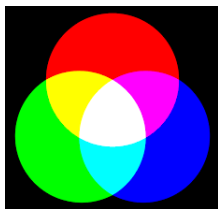
Con las características

```
FeaturesDict({  
    'image': Image(shape=(32, 32, 3), dtype=uint8),  
    'label': ClassLabel(shape=(), dtype=int64, num_classes=10),  
})
```




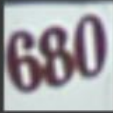

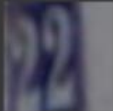
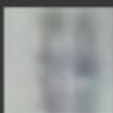
Datos

Imagenes: $(32,32,3) \rightarrow (32,32,1)$



Etiquetas: (0,1,2,3,4,5,6,7,8,9)

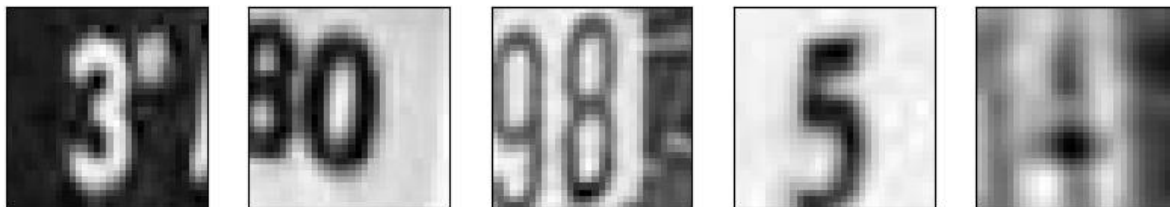
```
# Iteramos sobre todos los datos del conjunto de entrenamiento
for i, (imagen, etiqueta) in enumerate(datos['train']):
    # Redimensionamos la imagen a un arreglo de NumPy con el tamaño especificado
    imagen = cv2.resize(imagen.numpy(), (TAMANO_IMG, TAMANO_IMG))
    # Convertimos la imagen a escala de grises
    imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
    # Añadimos la imagen y su etiqueta a la lista de datos de entrenamiento
    datos_entrenamiento.append([imagen, etiqueta])
```

| | image | label |
|---|---|-------|
| 0 |  | 4 |
| 1 |  | 8 |
| 2 |  | 7 |
| 3 |  | 2 |
| 4 |  | 6 |

Cantidad de datos



Entrenamiento: 73,257



Validación: 26,032



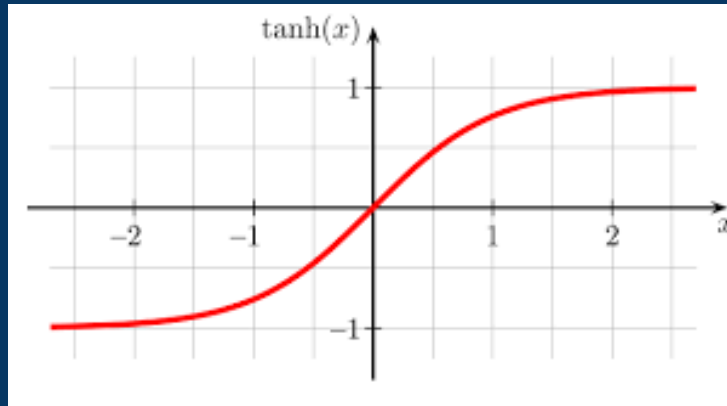
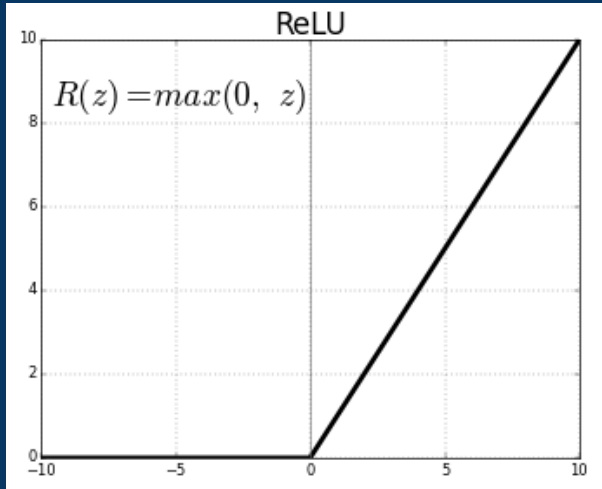
Prueba: 10,000

Modelos que vamos a entrenar

Modelo Denso

Modelo convolucional
Activación(relu)

Modelo convolucional
Activación(tanh)



Primer modelo:

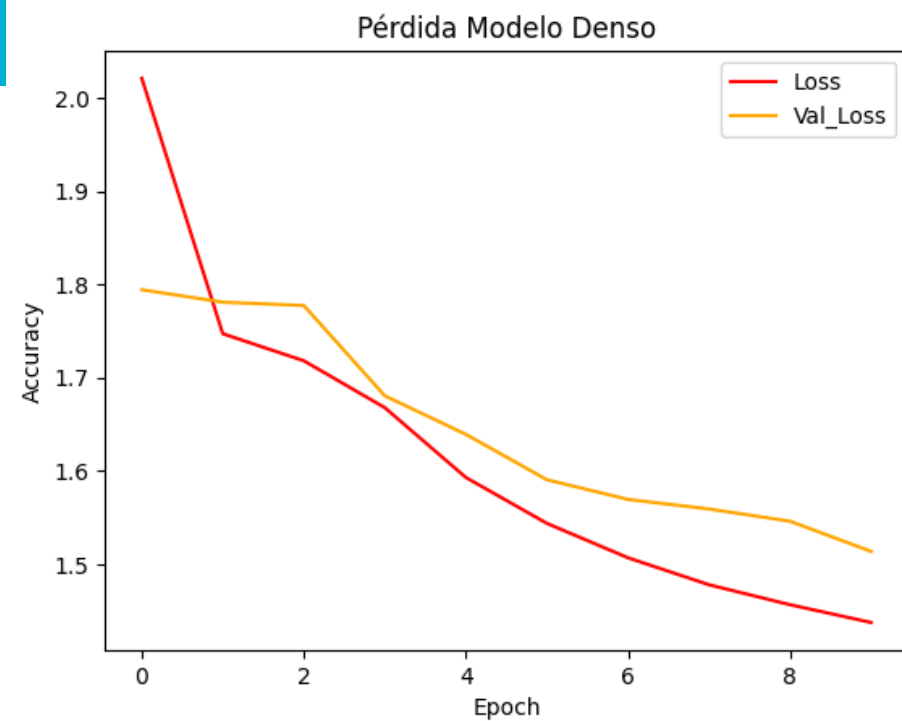
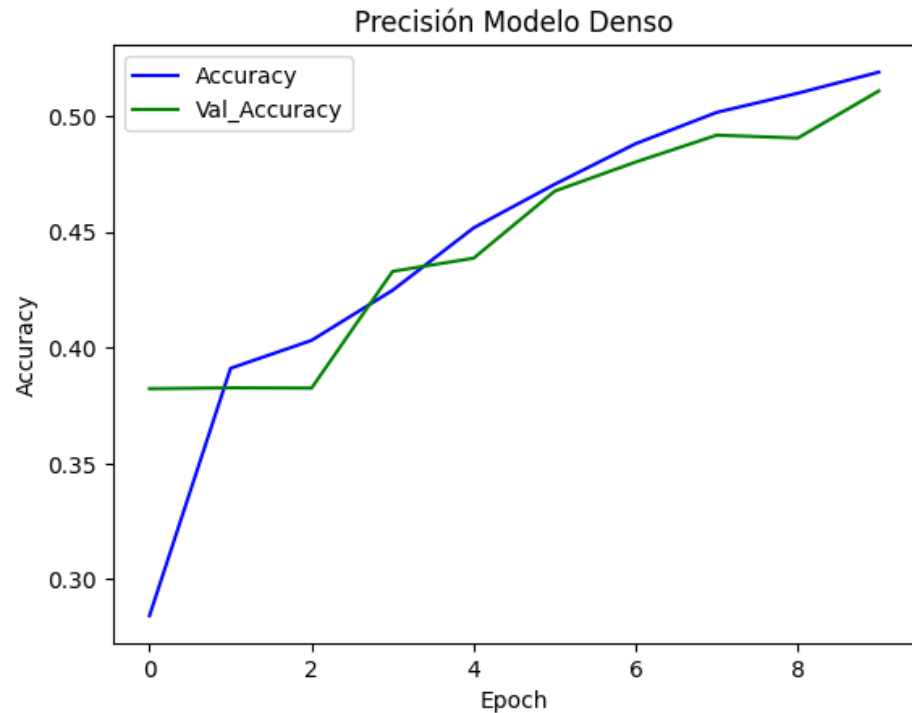
Red densa.

```
modeloDenso = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(32, 32, 1)), # Capa de aplanado  
    tf.keras.layers.Dense(64, activation='relu'), # Capa densa con activación ReLU  
    tf.keras.layers.Dense(64, activation='relu'), # Capa densa con activación ReLU  
    tf.keras.layers.Dense(10, activation='softmax') # Capa de salida con activación Softmax  
)
```

```
modeloDenso.compile(optimizer='adam',  
                    loss='sparse_categorical_crossentropy',  
                    metrics=['accuracy'])
```

```
histDenso = modeloDenso.fit(X, y,  
                            epochs=10,  
                            validation_data=(X_val, y_val),  
                            callbacks=[tensorboardDenso])
```

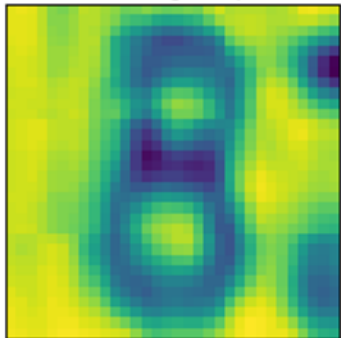
Tiempo de compilación: 2 min 25 s



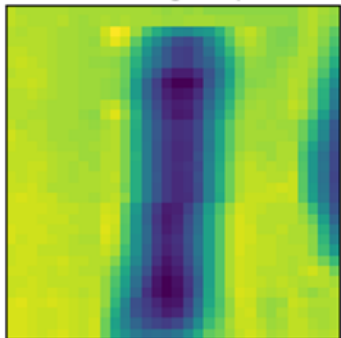
La perdida es: 1.2991520166397095
El accuracy es: 0.5569000244140625

Pruebas Modelo Denso

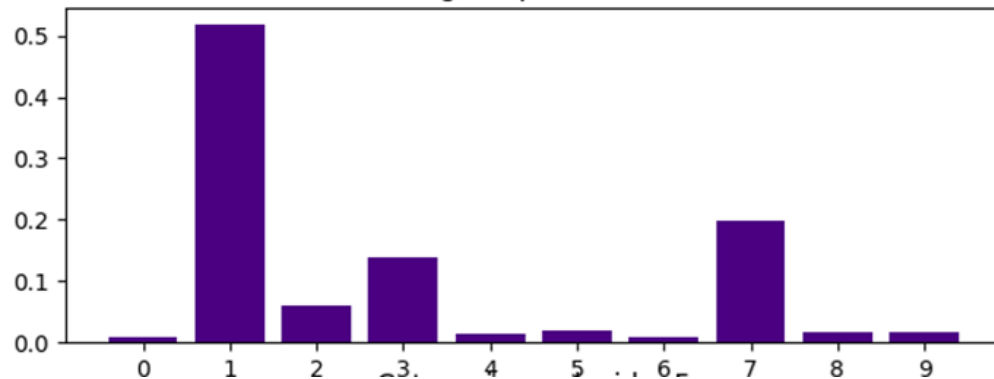
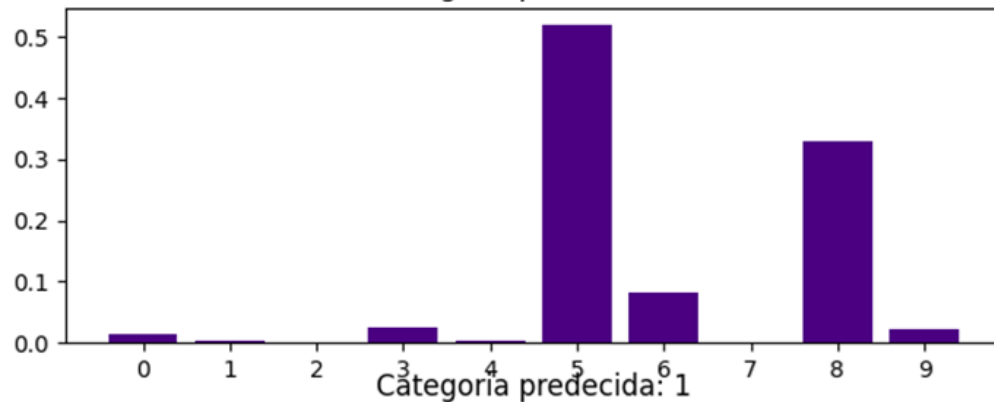
Categoría predecida: 8



Categoría predecida: 1



Categoría predecida: 5



Segundo modelo:

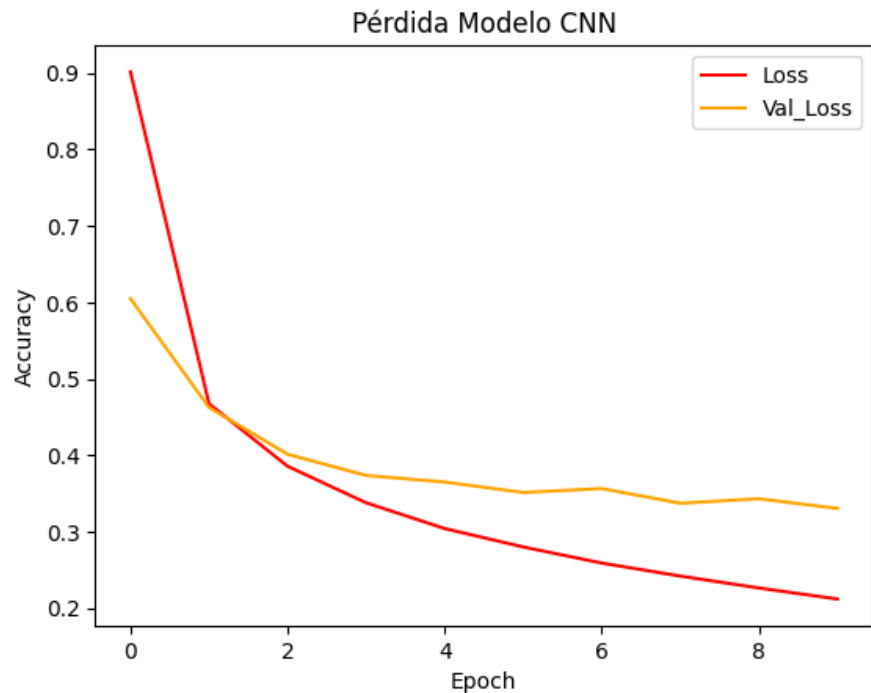
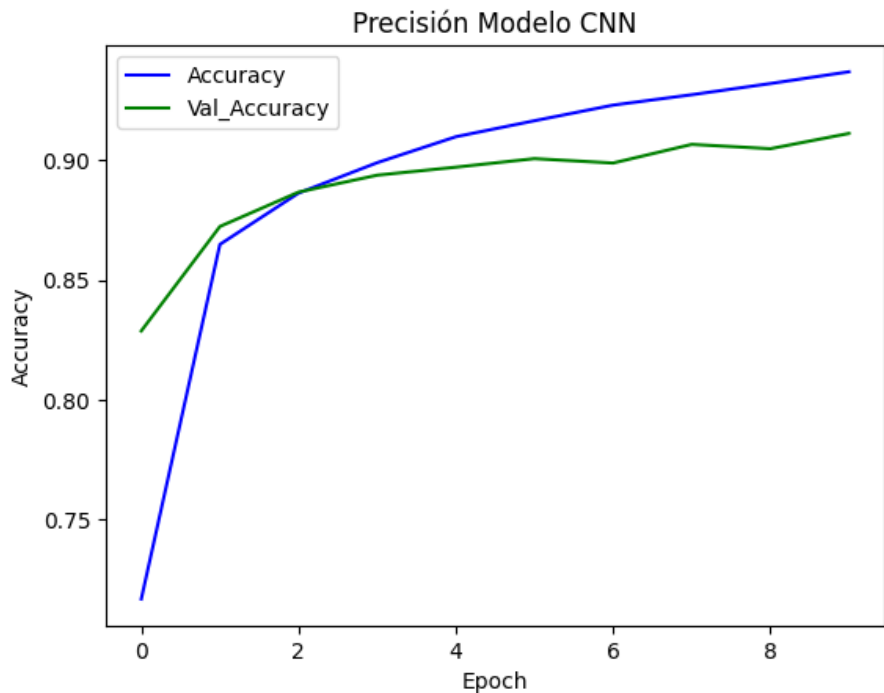
Modelo convolucional Activación(relu)

```
# Definir un modelo de red neuronal convolucional (CNN)
modeloCNN = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 1)), # Capa convolucional
    tf.keras.layers.MaxPooling2D(2, 2), # Capa de max pooling
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'), # Capa convolucional
    tf.keras.layers.MaxPooling2D(2, 2), # Capa de max pooling
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'), # Capa convolucional
    tf.keras.layers.MaxPooling2D(2, 2), # Capa de max pooling

    tf.keras.layers.Flatten(), # Capa de aplanado
    tf.keras.layers.Dense(100, activation='relu'), # Capa densa con activación ReLU
    tf.keras.layers.Dense(10, activation='softmax') # Capa de salida con activación Softmax
])
```

```
modeloCNN.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

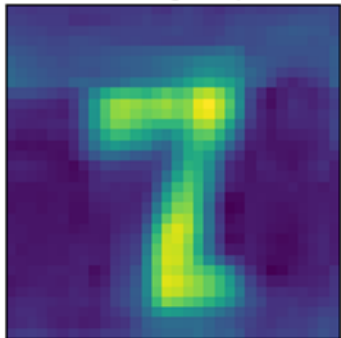
Tiempo de compilación: 2 min 23 s



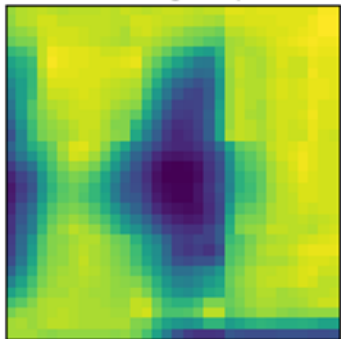
La perdida es: 0.17493025958538055
El accuracy es: 0.9516000151634216

Pruebas Modelo Convolucional (relu)

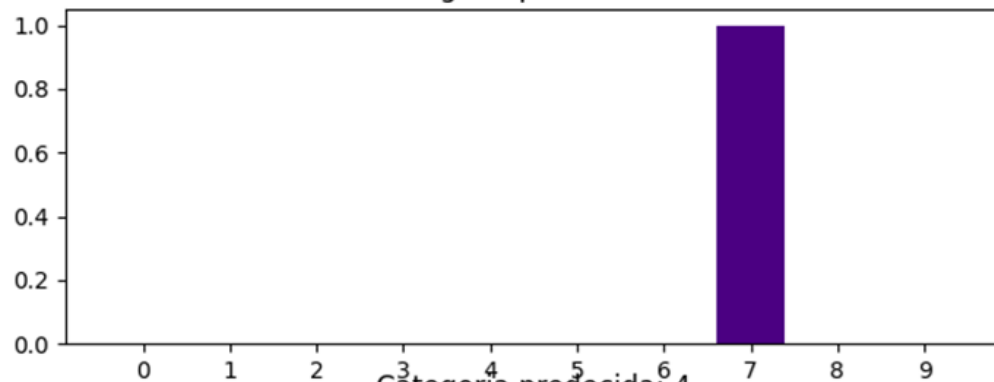
Categoría predecida: 7



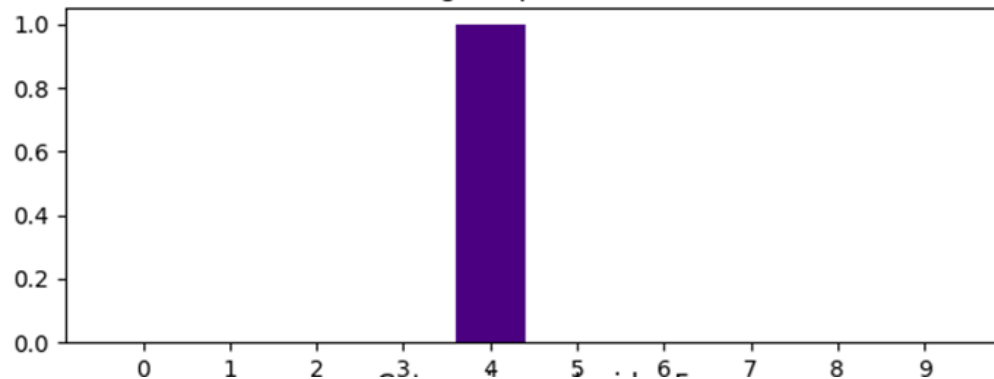
Categoría predecida: 4



Categoría predecida: 7



Categoría predecida: 4



Tercer modelo:

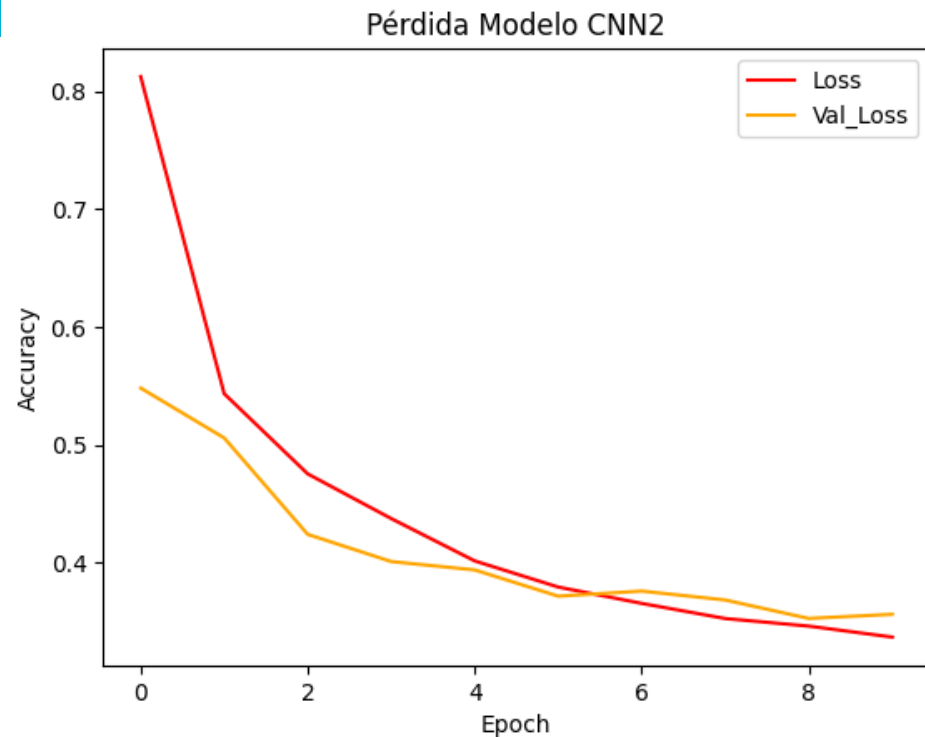
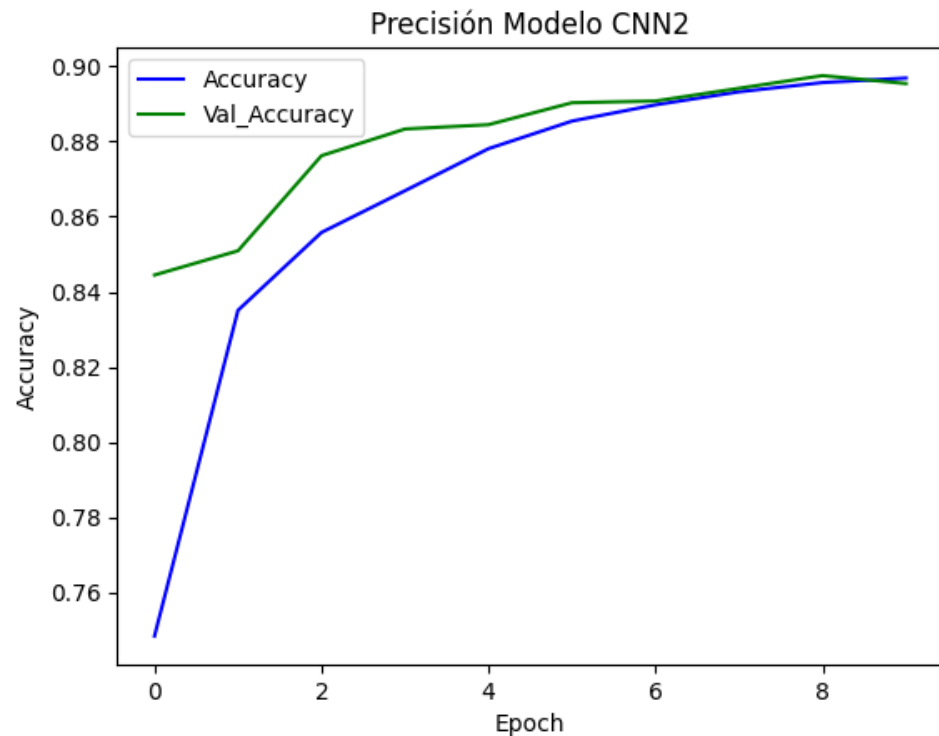
Modelo convolucional Activación(tanh)

```
"""Análogo al primer modelo convolucional, se genera una secuencia de capas con tensores como entradas,
en este caso se usan tangentes hiperbolicas (250) y una sigmoideal (10) como funciones de activacion y
haciendo uso de un dropout de 0.5"""
modeloCNN2 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='tanh', input_shape=(32, 32, 1)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='tanh'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3,3), activation='tanh'),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(250, activation='tanh'),
    tf.keras.layers.Dense(10, activation='sigmoid')
])
```

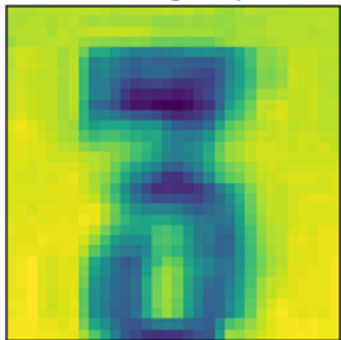
```
modeloCNN2.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

Tiempo de compilación: 2 min 23 s

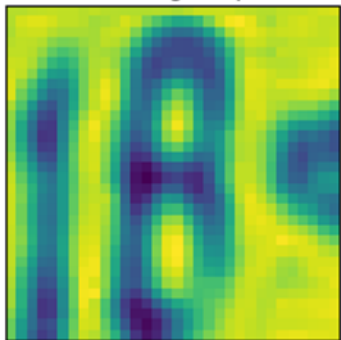


Pruebas Modelo Convolutional (tanh)

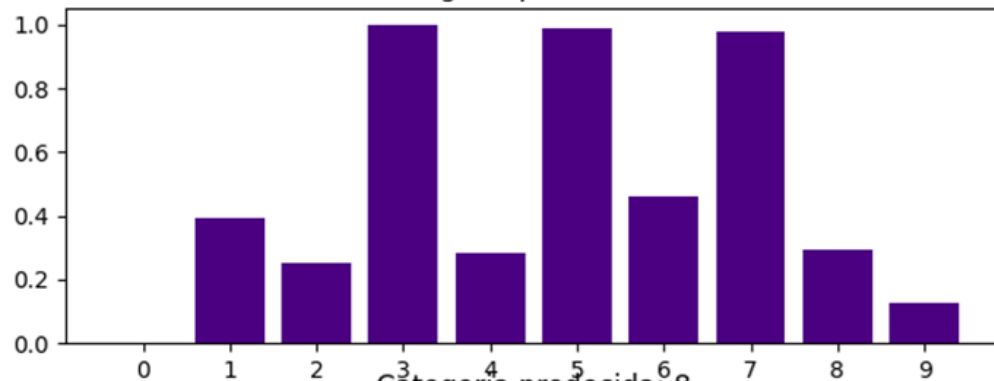
Categoría predecida: 3



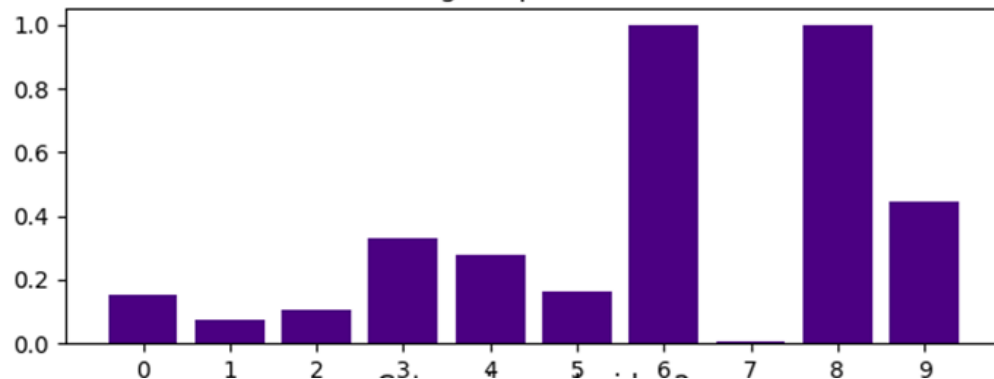
Categoría predecida: 8



Categoría predecida: 3



Categoría predecida: 8



Conclusiones

Precisión del modelo: El mejor modelo resultó ser la primera red convolucional definida. Esto debido a que es el que mejor evoluciona con función en las épocas vs costos y realiza las mejores predicciones con .

Desafíos específicos: Se trabajó el conjunto de datos, modificando su visualización para un mejor análisis.

Dificultades: La aplicación de los diferentes modelos sirve para identificar numeros unitarios, se tiene que trabajar en complementos para identificar adecuadamente un patrón de números de mas de una cifra.

Referencias

Artículo en donde se analizaron los datos por primera vez:

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng [Reading Digits in Natural Images with Unsupervised Feature Learning](#) *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. ([PDF](#))

Páginas donde se almacenan los datos :

<http://ufldl.stanford.edu/housenumbers>