Ciencia y Tecnología

Secretaría de Ciencia, Humanidades, Tecnología e Innovación

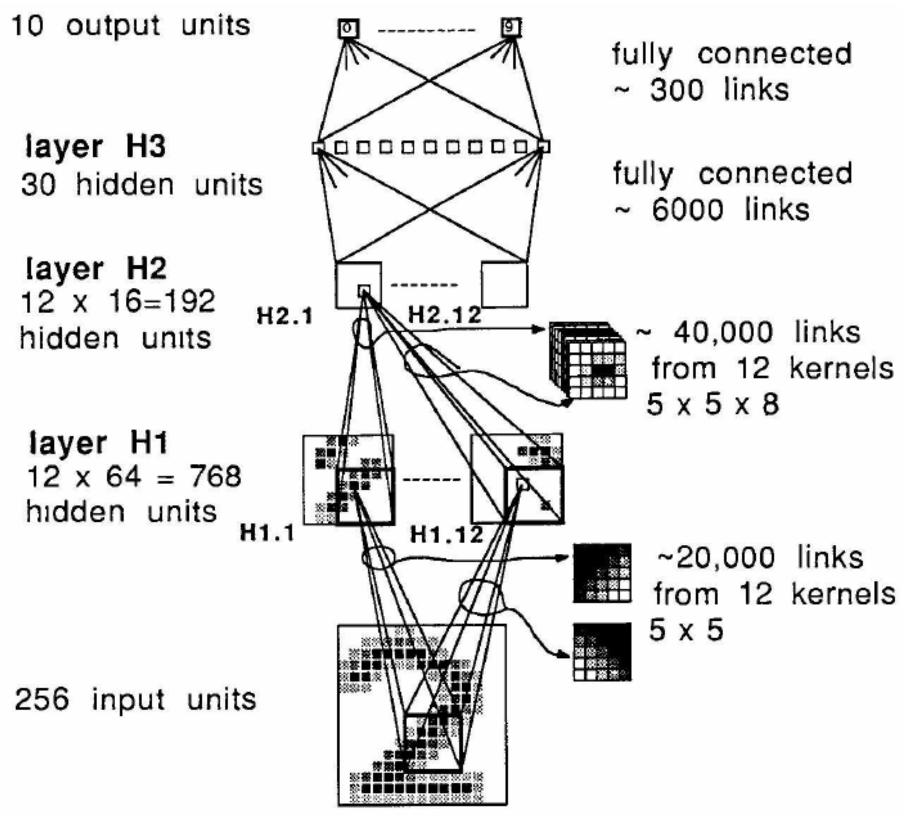
CIMAT
UNIDAD MÉRIDA

# REDES NEURONALES CONVOLUCIONALES

Dr. Francisco J. Hernández López
SECIHTI – CIMAT-Mérida
fcoj23@cimat.mx, www.cimat.mx/~fcoj23

# REDES NEURONALES CONVOLUCIONALES

- Diseñadas para el manejo de datos estructurados en forma de cuadrícula o malla, tales como una imagen.
- Las propiedades espaciales son heredadas de una capa a la siguiente.
- Son computacionalmente más eficientes que las redes neuronales complet. conect.

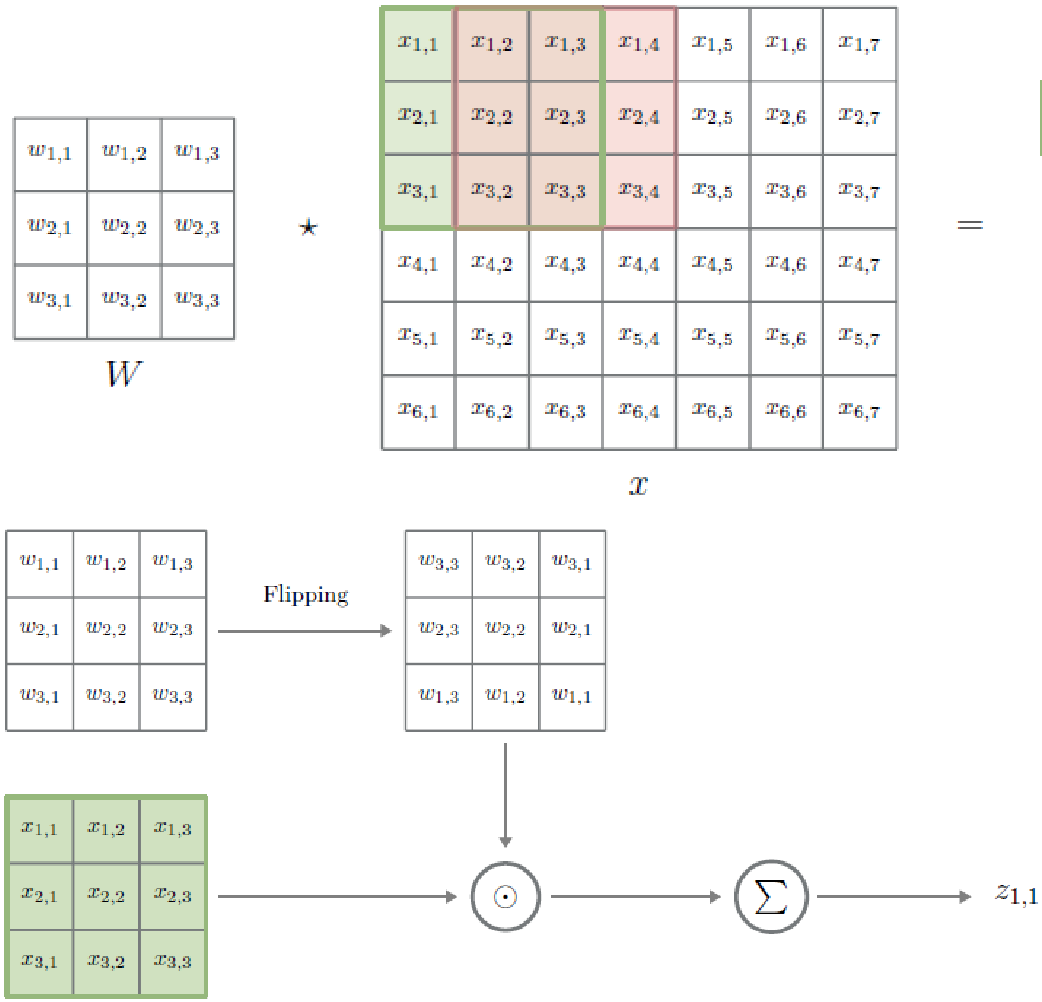




LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541-551.

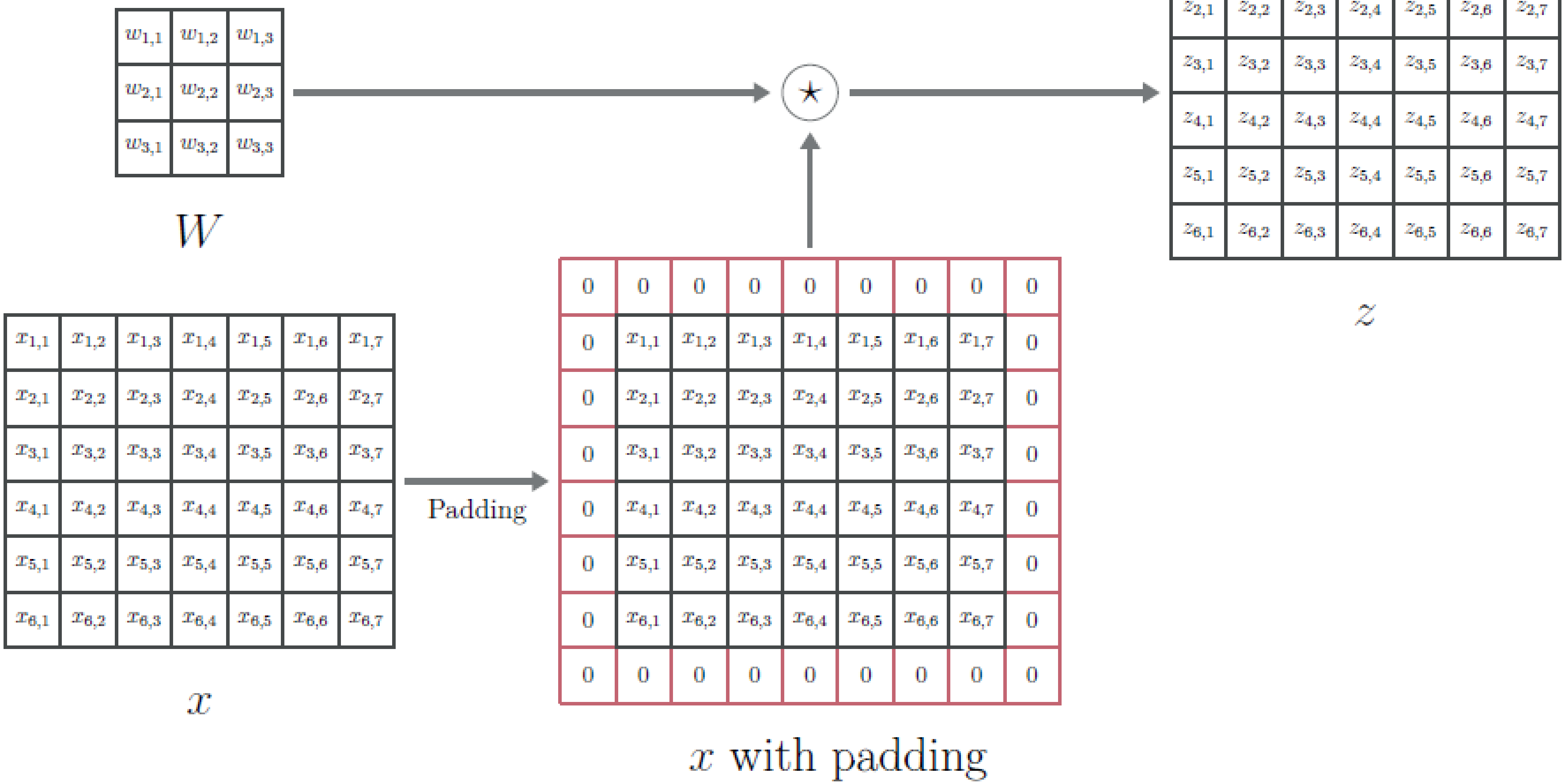
Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# CONVOLUCIÓN



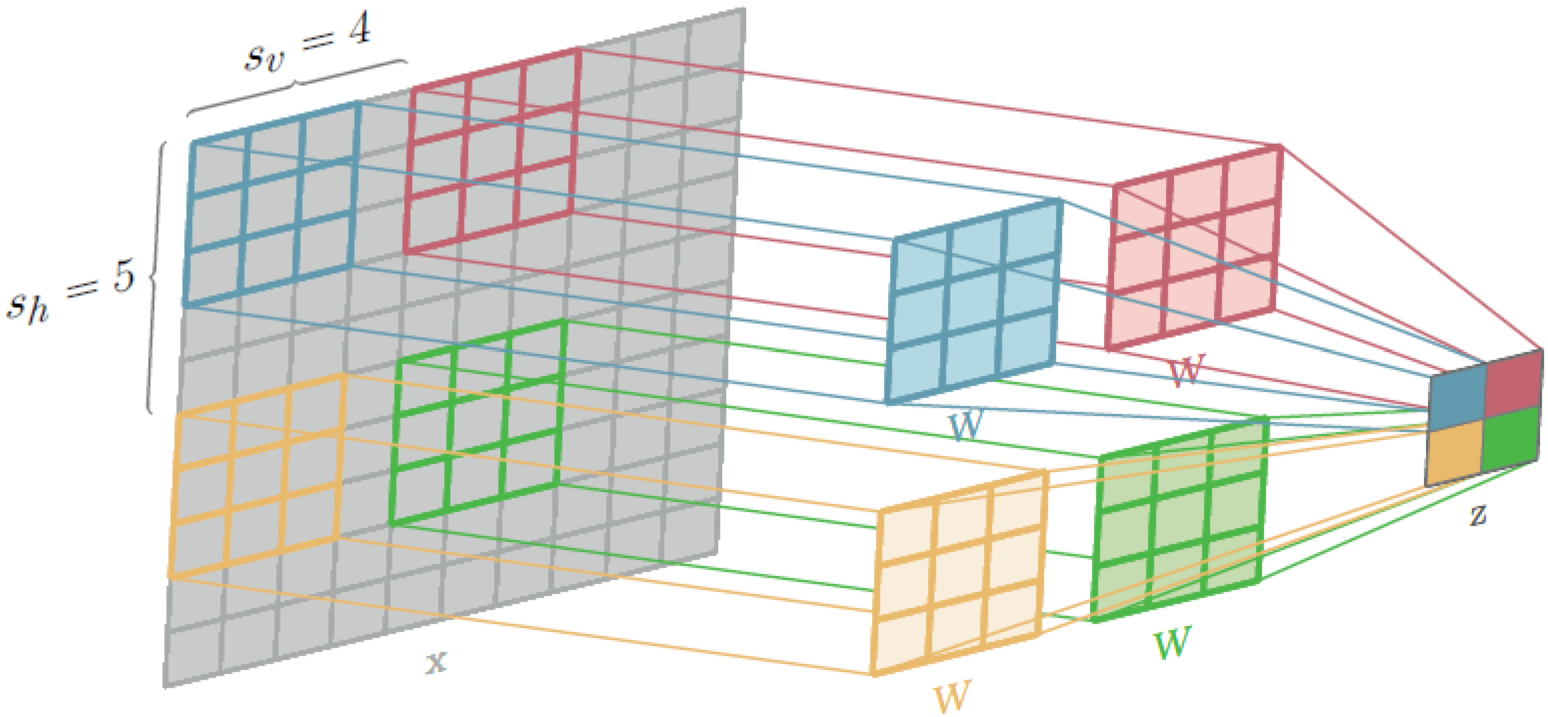

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# RELLENO (PADDING)



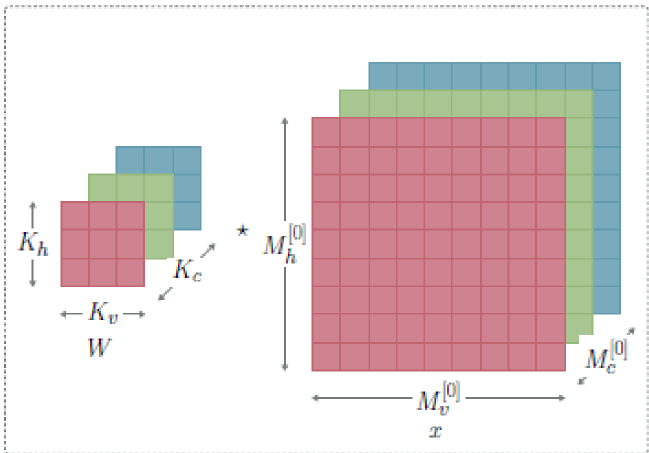

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

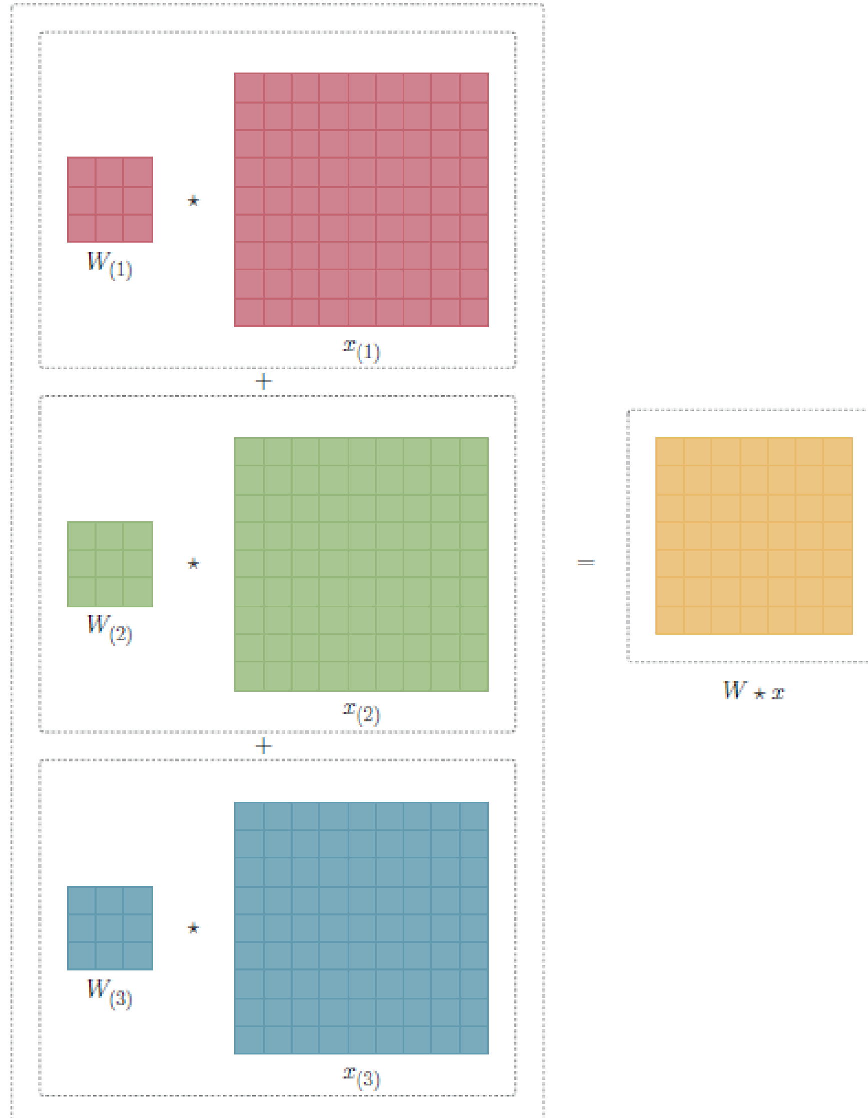
# ZANCADA (STRIDE)




Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# IMÁGENES RGB

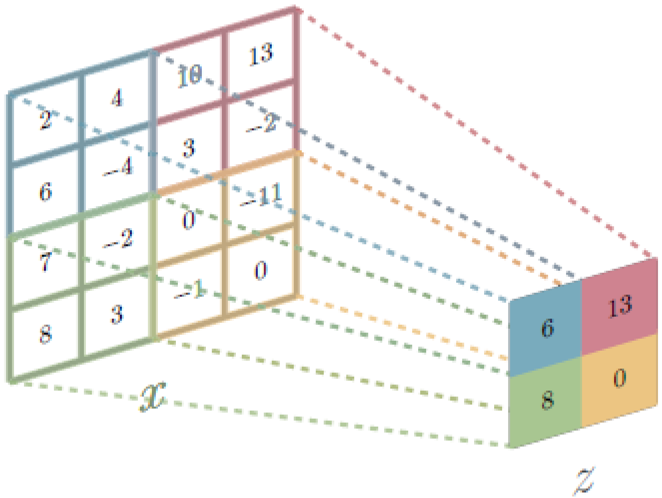


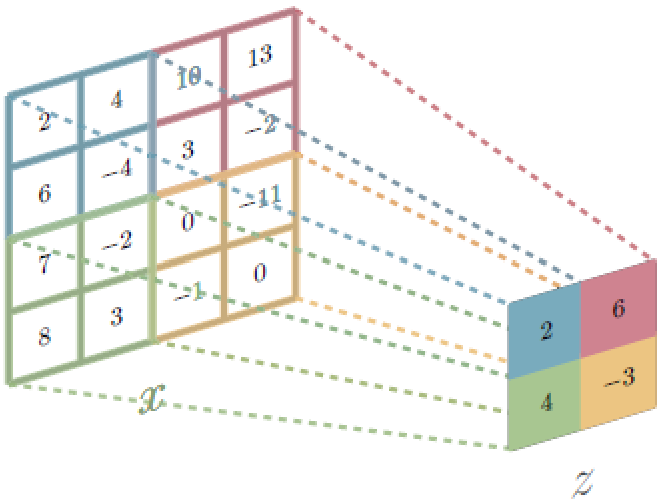

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# CAPAS DE AGRUPAMIENTO (POOLING)

- Son capas no entrenables.




Max-pooling



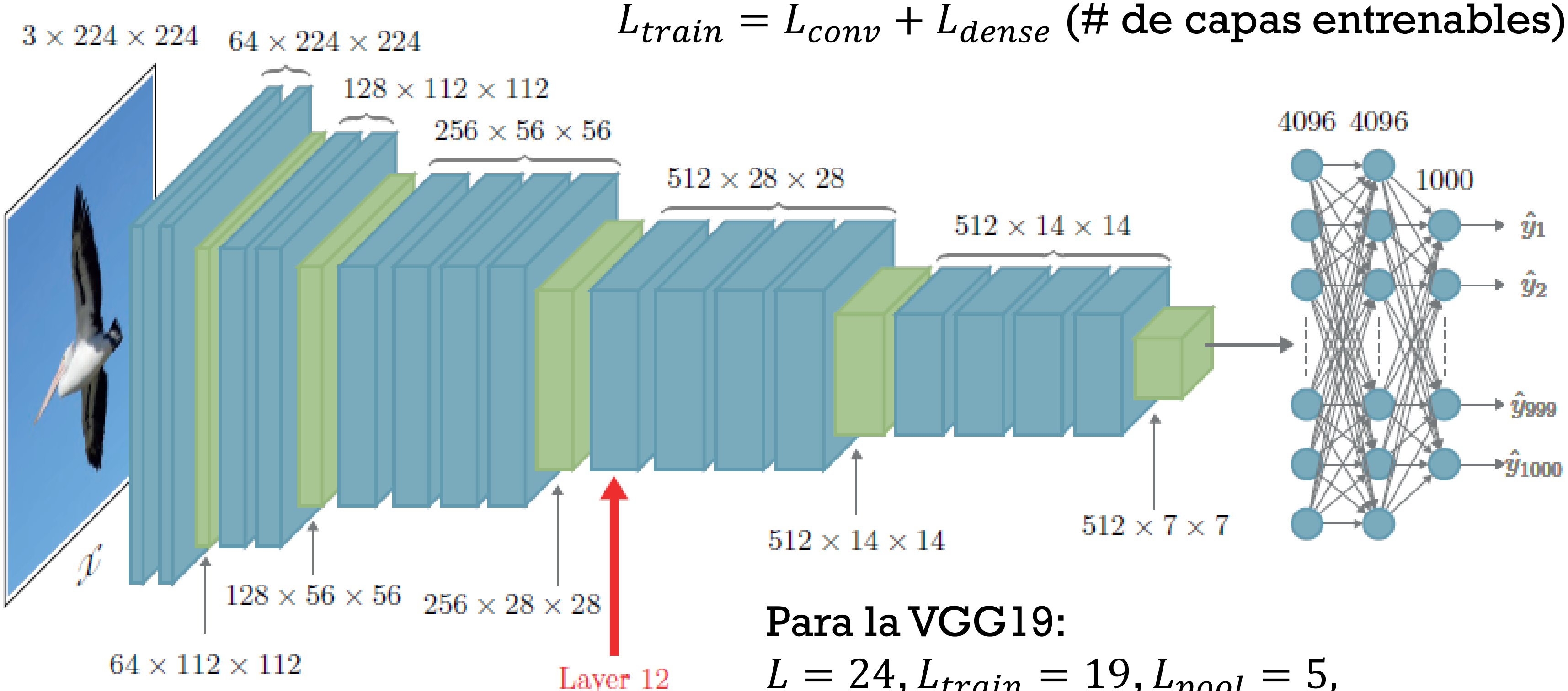

Average-pooling

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# RED NEURONAL CONVOLUCIONAL VGG19

$L = L_{train} + L_{pool}$

$L_{train} = L_{conv} + L_{dense}$ (# de capas entrenables)




**Para la VGG19:**

$L = 24, L_{train} = 19, L_{pool} = 5,$

$L_{conv} = 16$ **y** $L_{dense} = 3$


Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# CAPAS CONVOLUCIONALES

- Sea la capa $\ell$-ésima una capa convolucional, entonces

$$f_{\theta^{[\ell]}}^{(\ell)}\colon\ \mathbb{R}^{M_c^{[\ell-1]}\times M_h^{[\ell-1]}\times M_v^{[\ell-1]}} \to \mathbb{R}^{M_c^{[\ell]}\times M_h^{[\ell]}\times M_v^{[\ell]}}$$

- $f_{\theta^{[\ell]}}^{(\ell)}$ mapea $a^{[\ell-1]}$ a $a^{[\ell]}$

$$f_{\theta^{[\ell]}}^{(\ell)}\left(a^{[\ell-1]}\right) = S^{[\ell]}\left(\left[b_{(j)}^{[\ell]} + \sum_{i=1}^{M_c^{[\ell-1]}} W_{(j),(i)}^{[\ell]} \star a_{(i)}^{[\ell-1]}\right]_{j=1,\ldots,M_c^{[\ell]}}\right)$$

\# parámetros es:

$M_c^{[\ell]}\left(M_c^{[\ell-1]}K_h^{[\ell]}K_v^{[\ell]} + 1\right)$

$z_{(j)}^{[\ell]}$ es una matriz de $M_h^{[\ell]} \times M_v^{[\ell]}$

- El tensor de entrada tiene $M_c^{[\ell-1]}$ canales y el tensor de salida $M_c^{[\ell]}$ canales.

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# ARQUITECTURA VGG19

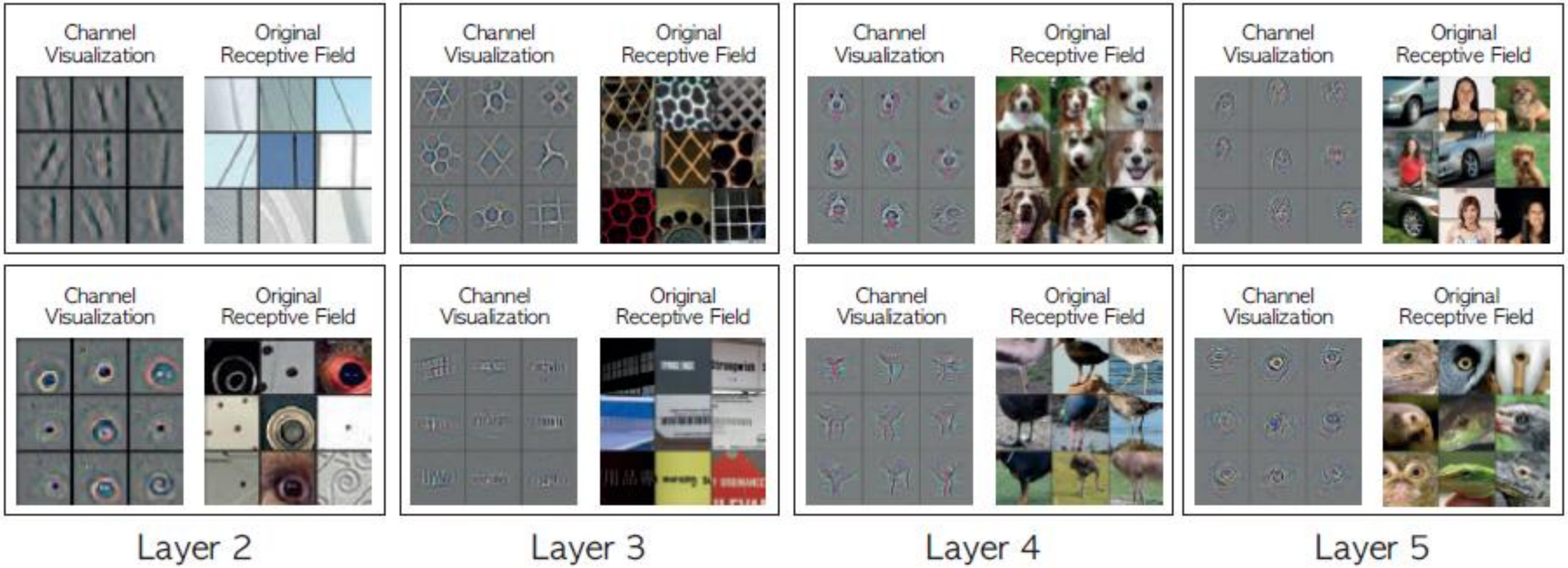
- $x$ es de $M_c^{[0]} \times M_h^{[0]} \times M_v^{[0]}$.
- $K = 1000$ clases.
- Kernels de conv. de $3 \times 3$.
- Padding $(p_h, p_v) = (2{,}2)$.
- Stride $(s_h, s_v) = (1{,}1)$.
- Max-pooling de $2 \times 2$.
- # de canales se incrementa de 3 a 512.
- Función de activ.: ReLU.
- Soft-max para la capa de salida

| Layer Number | Type of layer | Output dimension | Number of neurons | Number of learned parameters |
|---|---|---|---|---|
| 0 | Input | $3 \times 224 \times 224$ | - | - |
| 1 | Convolution | $64 \times 224 \times 224$ | 3,211,264 | 1,792 |
| 2 | Convolution | $64 \times 224 \times 224$ | 3,211,264 | 36,928 |
| 3 | Max-pooling | $64 \times 112 \times 112$ | 802,816 | 0 |
| 4 | Convolution | $128 \times 112 \times 112$ | 1,605,632 | 73,856 |
| 5 | Convolution | $128 \times 112 \times 112$ | 1,605,632 | 147,584 |
| 6 | Max-pooling | $128 \times 56 \times 56$ | 401,408 | 0 |
| 7 | Convolution | $256 \times 56 \times 56$ | 802,816 | 295,168 |
| 8 | Convolution | $256 \times 56 \times 56$ | 802,816 | 590,080 |
| 9 | Convolution | $256 \times 56 \times 56$ | 802,816 | 590,080 |
| 10 | Convolution | $256 \times 56 \times 56$ | 802,816 | 590,080 |
| 11 | Max-pooling | $256 \times 28 \times 28$ | 200,704 | 0 |
| 12 | Convolution | $512 \times 28 \times 28$ | 401,408 | 1,180,160 |
| 13 | Convolution | $512 \times 28 \times 28$ | 401,408 | 2,359,808 |
| 14 | Convolution | $512 \times 28 \times 28$ | 401,408 | 2,359,808 |
| 15 | Convolution | $512 \times 28 \times 28$ | 401,408 | 2,359,808 |
| 16 | Max-pooling | $512 \times 14 \times 14$ | 100,352 | 0 |
| 17 | Convolution | $512 \times 14 \times 14$ | 100,352 | 2,359,808 |
| 18 | Convolution | $512 \times 14 \times 14$ | 100,352 | 2,359,808 |
| 19 | Convolution | $512 \times 14 \times 14$ | 100,352 | 2,359,808 |
| 20 | Convolution | $512 \times 14 \times 14$ | 100,352 | 2,359,808 |
| 21 | Max-pooling | $512 \times 7 \times 7$ | 25,088 | 0 |
| Flattening to a vector of length 25,088 | | | | |
| 22 | Fully connected | 4,096 | 4,096 | 102,764,544 |
| 23 | Fully connected | 4,096 | 4,096 | 16,781,312 |
| 24 | Fully connected | 1,000 | 1,000 | 4,097,000 |
| | | | Total: 16,391,656 | Total: 143,667,240 |

Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# VISUALIZACIÓN

- Una representación visual podría ayudar a entender la función que tienen los canales individuales en la red.




Liquet, B., Moka, S., & Nazarathy, Y. (2024). Mathematical Engineering of Deep Learning (1st ed.). Chapman and Hall/CRC.

# EJEMPLO: MNIST USANDO UNA CNN

Intro_CNN_MNIST.py > ...

```python
#Ejemplo: Simple CNN para el problema del MNIST
#Mis ambientes: env_pi38_tf25 (Python 3.8 con TensorFlow 2.5)

# Importing libraries
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.utils import plot_model

# Loading the MNIST dataset
path = 'D:/fcoj23/CIMAT_MERIDA/Investigacion/objectDetection/deepLearning/CNNs/mnist.npz'
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data(path)
```

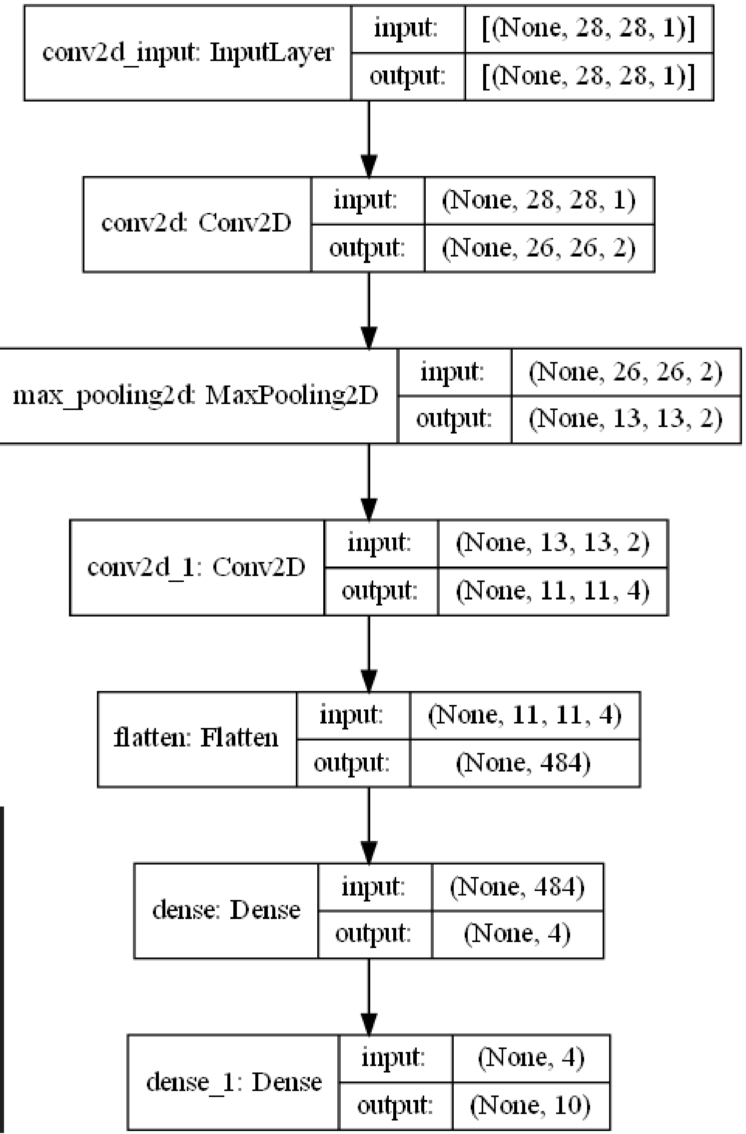
```python
# Preprocessing the data
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
```

# MODELO V1

```
# Building the CNN model
model = models.Sequential([
    layers.Conv2D(2, (3, 3), activation='relu',
                  input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(4, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(4, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
plot_model(model,show_shapes=True,
           to_file='model_basic_CNN.png')

# Compiling the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

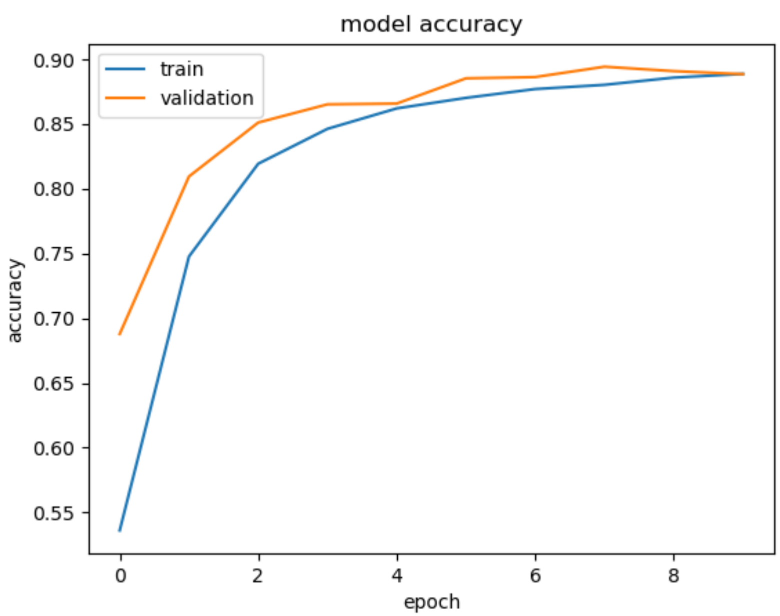


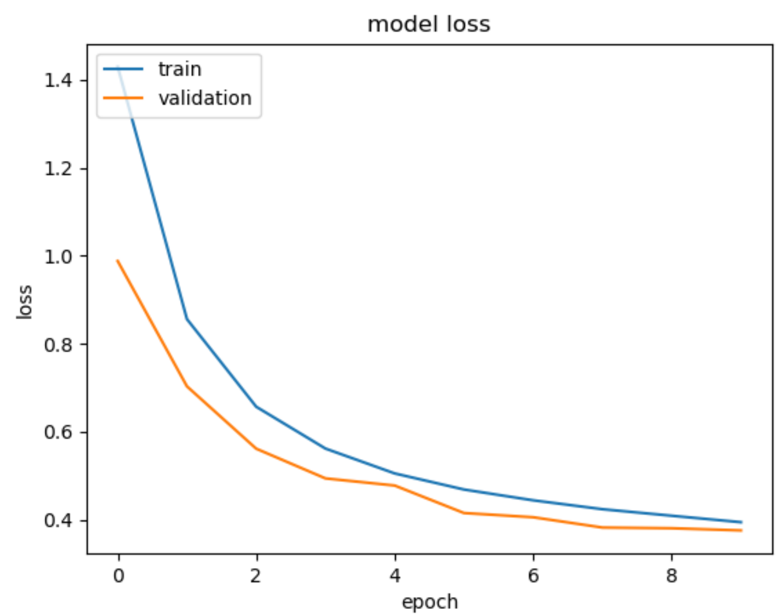
# ENTRENAMIENTO V1

```python
# Training the model
history = model.fit(train_images, train_labels,
                    epochs=10, batch_size=64,
                    validation_split=0.2)

# Evaluating the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_acc}')
```

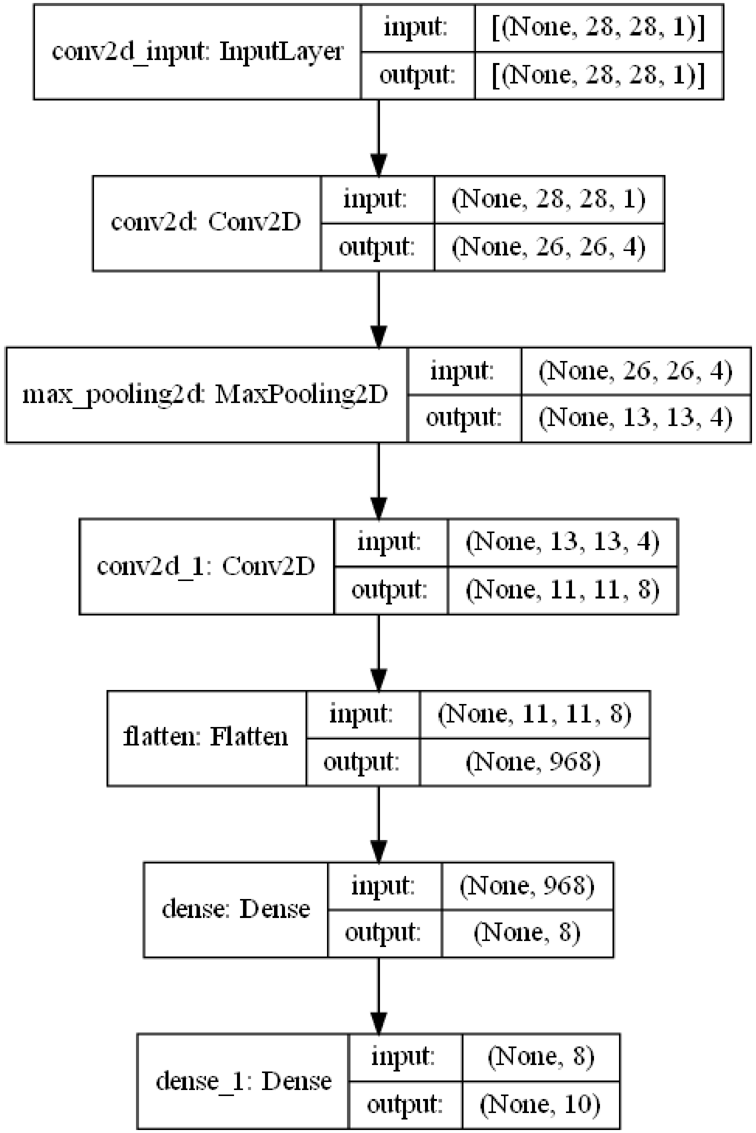
```
Test Loss: 0.4010499119758606
Test Accuracy: 0.8834999799728394
```



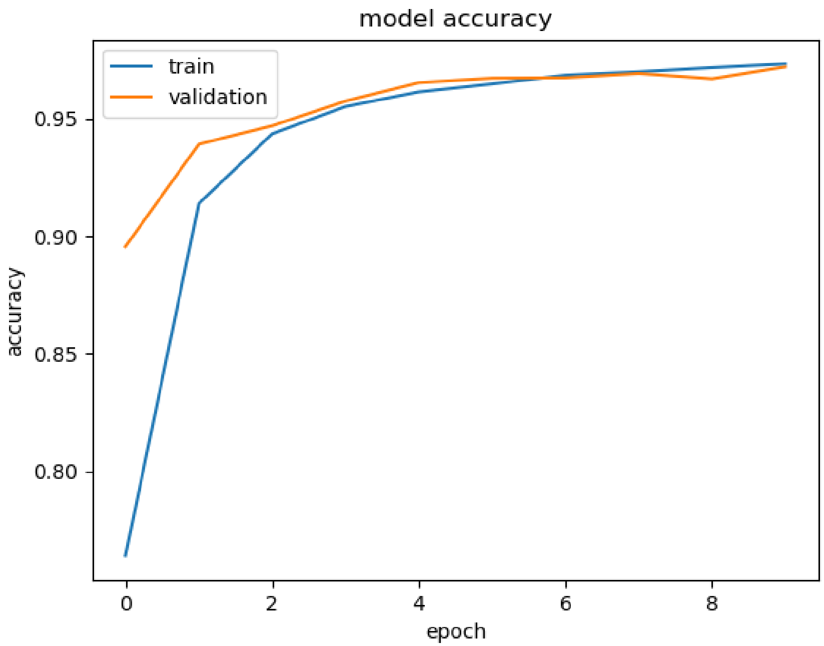



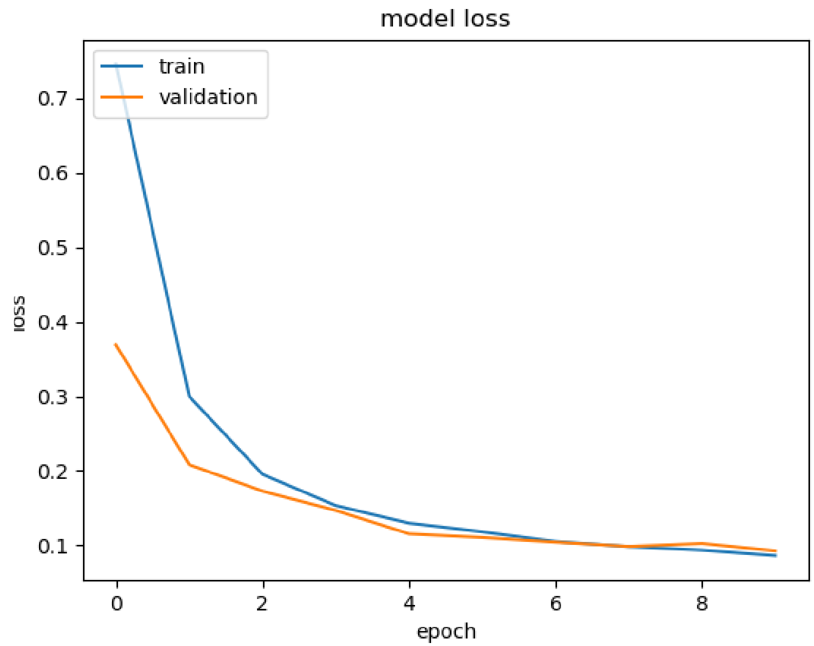
# MODELO V2

```python
# Building the CNN model
model = models.Sequential([
    layers.Conv2D(4, (3, 3), activation='relu',
                  input_shape=(28, 28, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(8, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(8, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```



# ENTRENAMIENTO V2




```
Test Loss: 0.08737307786941528
Test Accuracy: 0.9739000201225281
```

# GRACIAS POR SU ATENCIÓN

Francisco J. Hernández López

fcoj23@cimat.mx

WebPage:

www.cimat.mx/~fcoj23