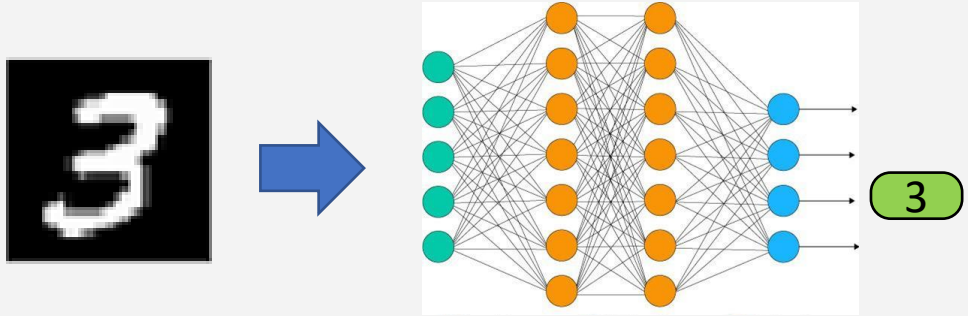
# Clasificación de Imágenes

# Clasificación de imágenes

Para crear un modelo que permita clasificar imágenes es necesario tener en cuenta algunas consideraciones:

- Las imágenes tienen los pixeles en formato matricial. Un modelo neuronal no acepta esto. Es necesario “aplanar” la imagen.
- Si es una imagen color, la cantidad de pixeles se triplica al tener los 3 canales.



# Clasificación de imágenes

Imagen original

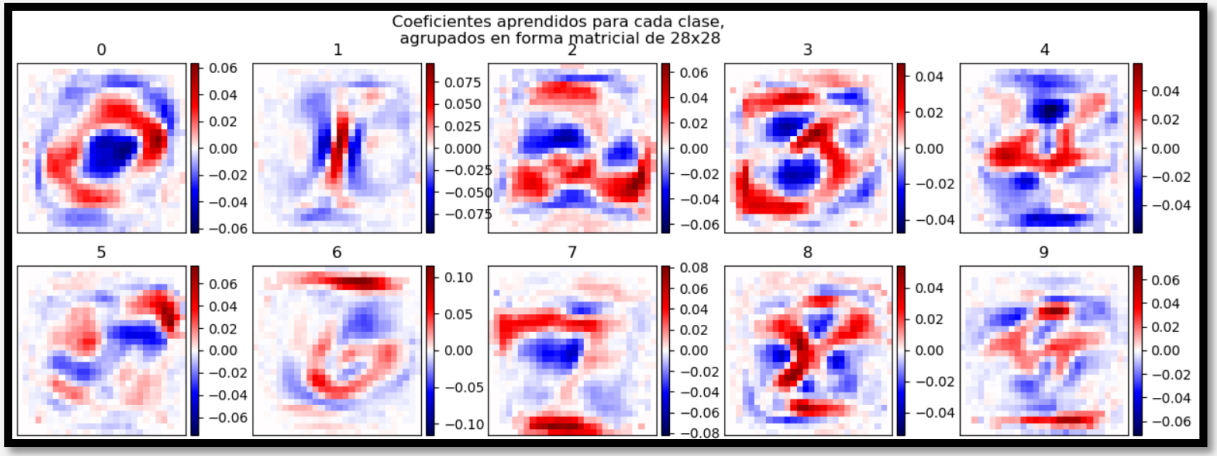

Aplanar

Imagen aplanada

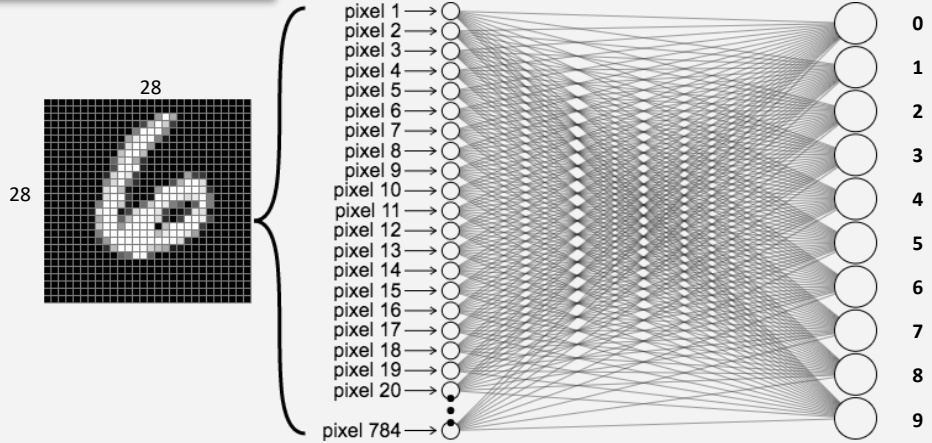

```
# Reshape de una imagen individual
img = np.reshape(img, (img.shape[0]* img.shape[1]))
# Reshape de todo el dataset
X = np.reshape(X, (X.shape[0], X.shape[1]*X.shape[2]))
```

```
# En Keras se agrega una capa “flatten” al comienzo para que aplane la imagen.
model.add(Flatten(input_shape= X_train[0].shape))
```

# Clasificación de imágenes - MNIST




Regresión Logística



# Clasificación de imágenes – CIFAR10

CIFAR10 es una base de datos mucho más realista que contiene decenas de miles de imágenes con diferentes objetos como animales y vehículos.

Cada imagen está codificada en 32x32 pixeles con 3 canales de color.

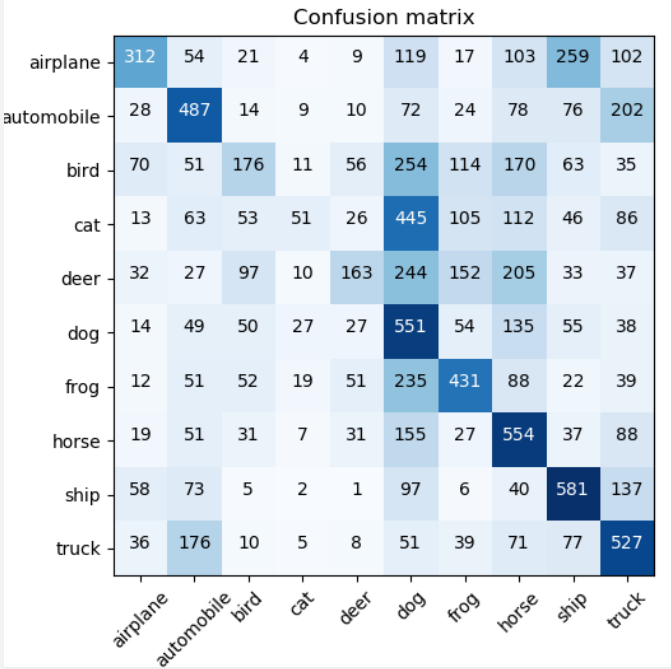
# Clasificación de imágenes – CIFAR10

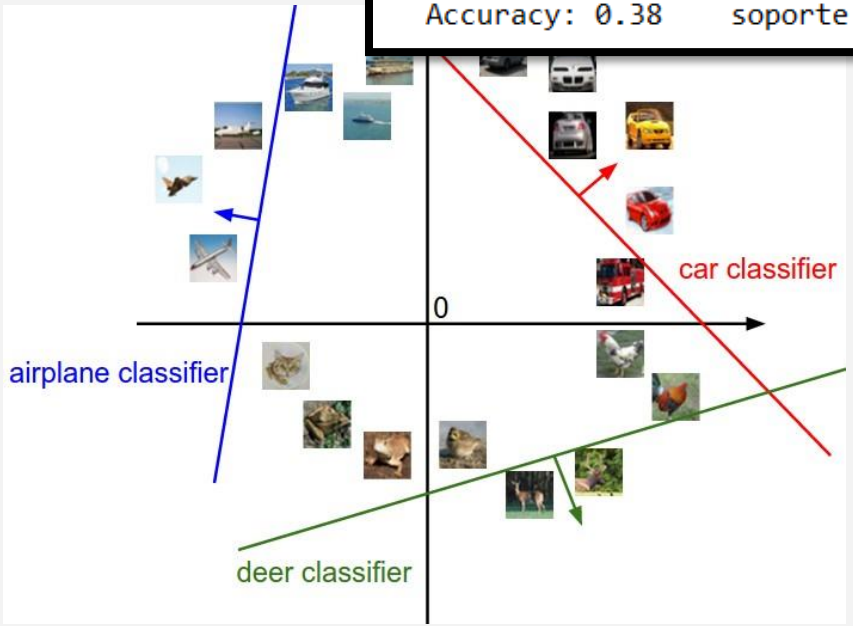
**38%** de Acc es mejor que 1/10 = **10%** (tirar una moneda).

Estos resultados son esperables ya que no es un problema simple. Las imágenes son muy distintas y con mucho ruido.

```
Train
   Accuracy: 0.42     soporte: 50000

Test
   Accuracy: 0.38     soporte: 10000
```

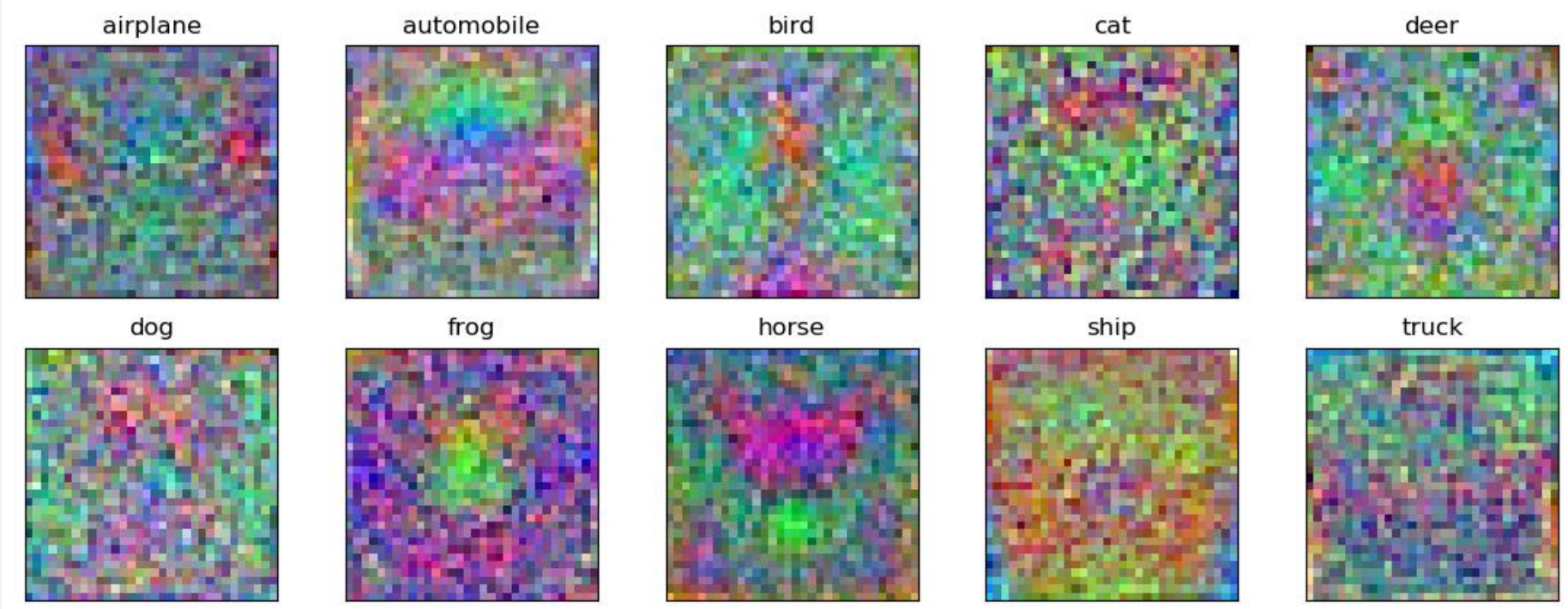


# Clasificación de imágenes – CIFAR10

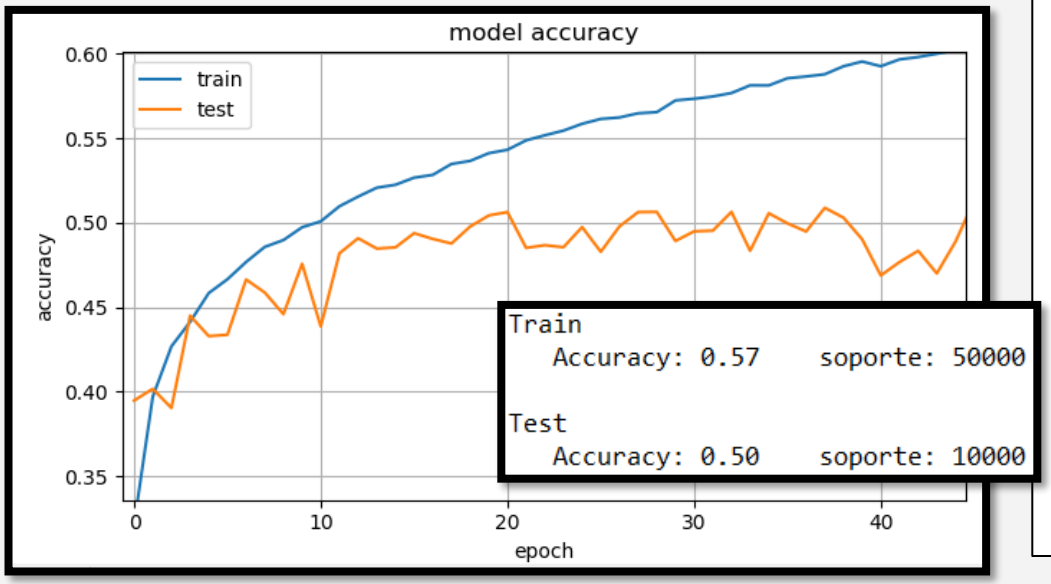
Vector de pesos aprendidos por cada clase (graficados en forma de matriz).

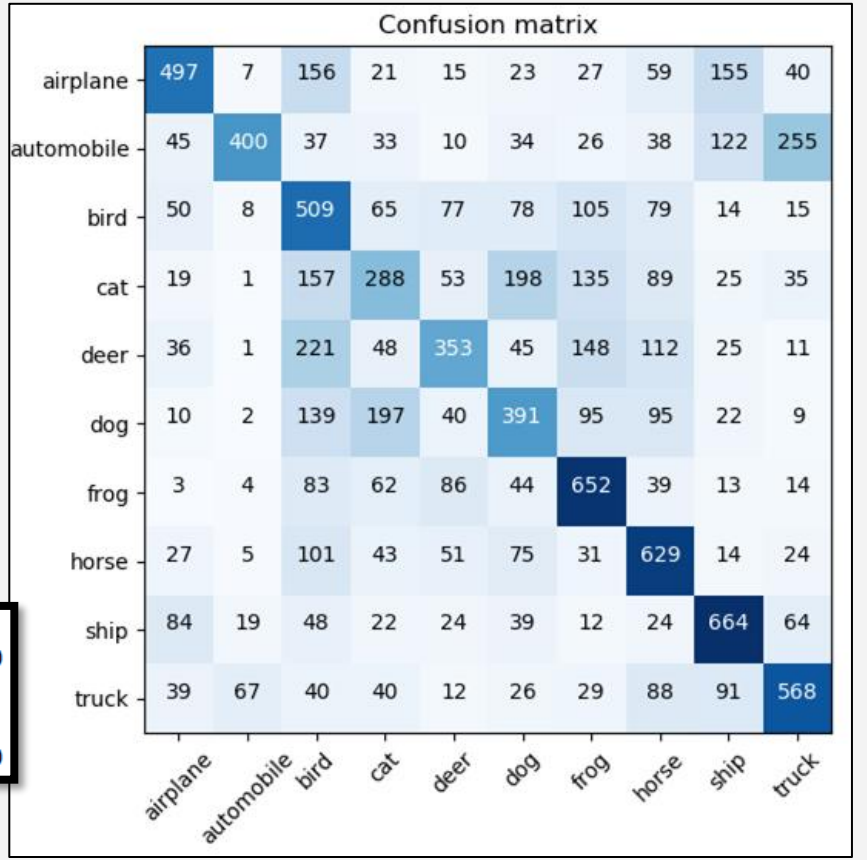
```
w_r= np.reshape(w, (32,32,3,10))
```



# Clasificación de imágenes – CIFAR10

Al agregar una capa oculta a la red elevamos el Acc a 50% (para el testing set). Cuantas más capas ocultas, corremos riesgo de caer en Overfitting.



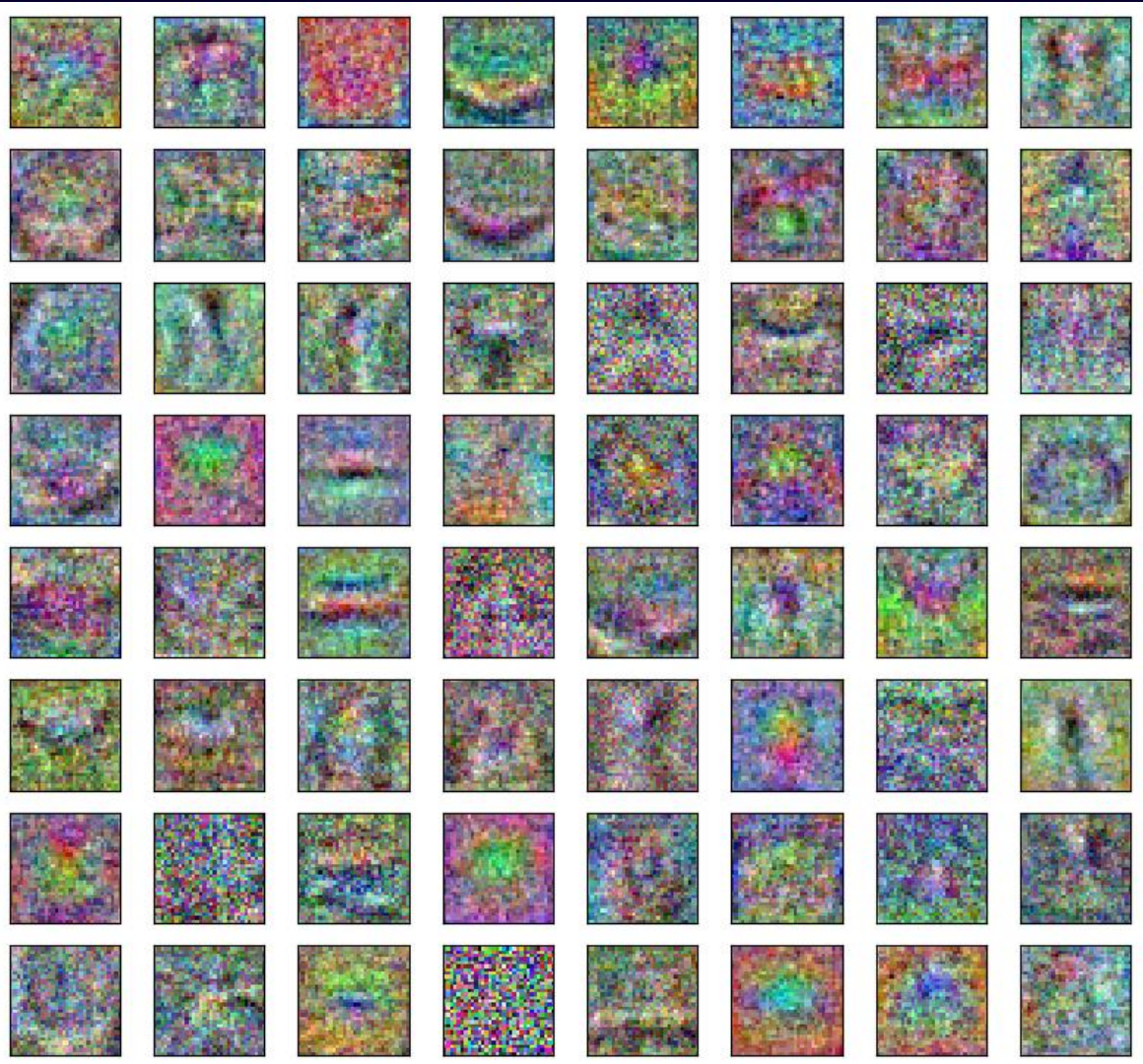



**Arquitectura**

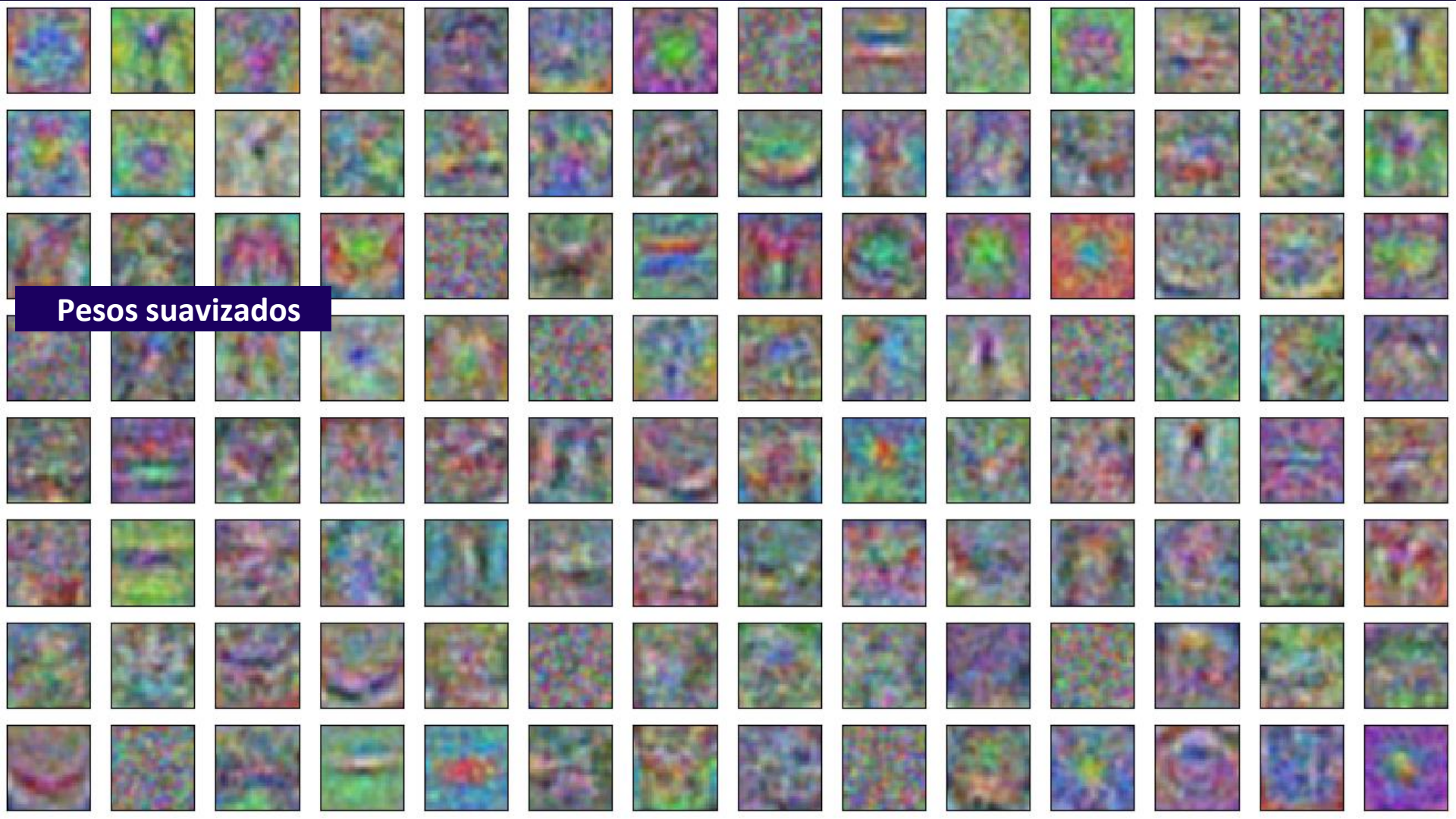
```
Flatten()
Dense(64, ‘Relu’)
Dense(10, ‘softmax’)
```

Pesos de las **64** neuronas de la capa oculta.

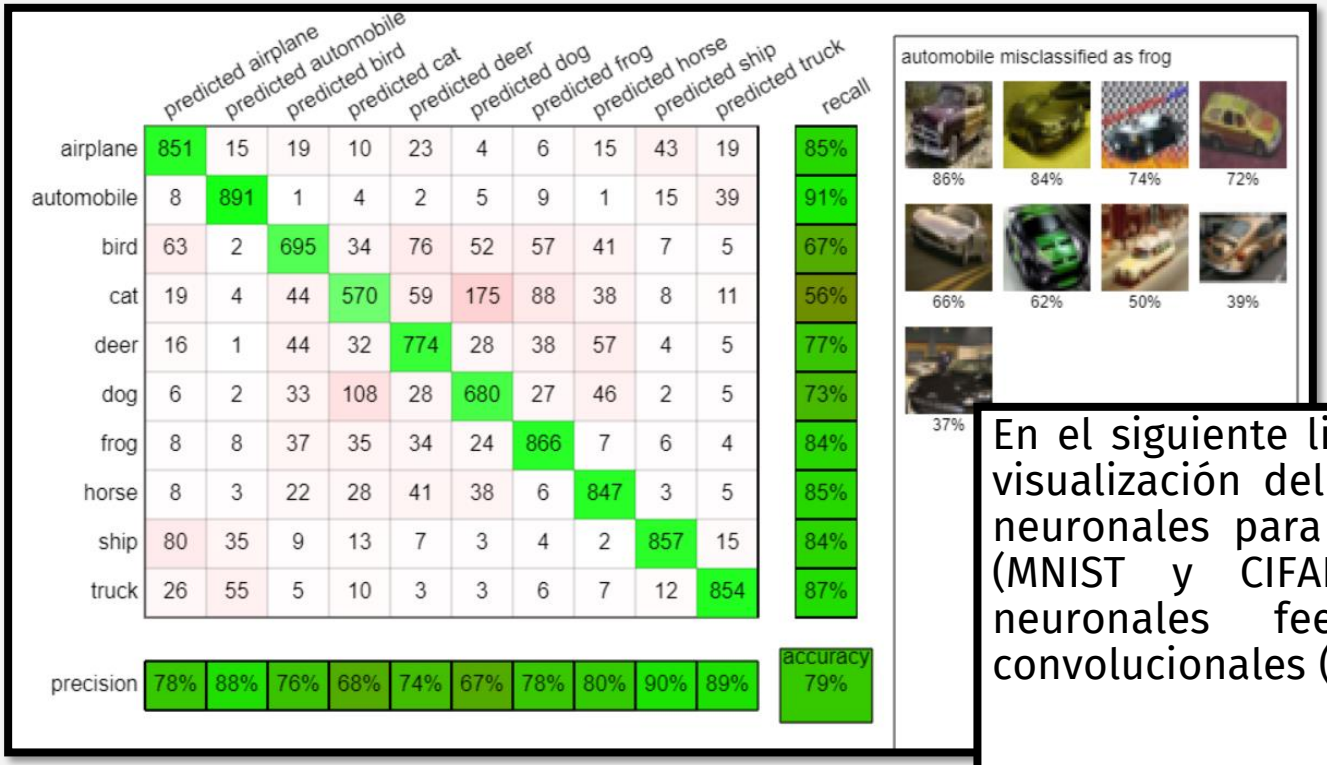
Notar que pareciera que algunas neuronas “encuentran” ciertos patrones en las imágenes.




112 neuronas ocultas


Pesos suavizados

# Matriz de confusión




En el siguiente link podrán encontrar una visualización del entrenamiento de redes neuronales para las dos bases de datos (MNIST y CIFAR10), tanto con redes neuronales feed-forward como con convolucionales (próxima clase).

https://ml4a.github.io/demos/confusion_mnist/