



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Reconocimiento de Formas y Aprendizaje Computacional

Trabajo Academico

Luis Alfonso Cardoza Bird

17 de Octubre de 2023

Cifar 100

Introduccion

Utilizando como base el **dataset** de **CIFAR-100**, se haran diversos procesos de clasificación de imágenes.

Objetivo

- Proveer un **dataset** standard, que compare clasificadores de imágenes.

Importando Datos / Etiquetas

```
(X_train, Y_train), (X_test, Y_test) = cifar100.load_data()
```

Downloading data from <https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz>
169001437/169001437

Se requiere tener una base de datos para poder efectuar nuestro análisis.

```
fine_labels = [  
    'apple',  
    'aquarium_fish',  
    'baby',  
    ...  
]  
for instances in fine_labels:  
    class_names.update({counter: instances})  
    counter+=1
```

Se realiza una iteración para poder obtener los etiquetados que van a ser utilizados para nuestras clasificaciones.

Escaneado de Imágenes

```
from random import randint

fig_display = plt.figure(figsize=[10, 10])
chosen_idx = randint(0, 50000-21)

for indx in range(chosen_idx, chosen_idx+20, 1):
    axes = fig_display.add_subplot(4, 5, indx - chosen_idx +1)
    axes.imshow(X_train[indx, :, :])
    axes.set_xticks([ ])
    axes.set_yticks([ ])
    axes.set_title("Class: {}".format(class_names[Y_train[indx,0]]))
```

Class: beaver



Class: ray



Class: shrew



Class: pear



Class: cockroach



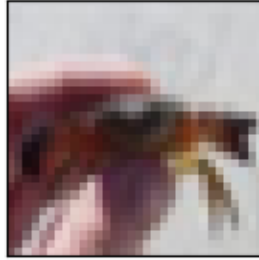
Class: streetcar



Class: tank



Class: crab



Class: bridge



Class: motorcycle



Class: fox



Class: tank



Class: lawn_mower



Class: couch



Class: forest



Class: whale



Class: lizard



Class: dinosaur



Class: chair



Class: cloud



Categorizacion de Etiquetas

Hace referencia a la transformación de las etiquetas por categorías previamente mencionadas con el dataset de imanes para dar lugar a un formato que pueda procesar y comprender eficientemente dicha clasificación.

Es necesario utilizar el metodo **One-hot encoding** para obtener los mejores resultados.

One hot

```
onehot_train = to_categorical(Y_train)
onehot_test = to_categorical(Y_test)

print(onehot_train[:2])
```

```
[
  [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
  [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
    0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]
]
```

Estructura

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 32, 32, 32)	896
conv2d_9 (Conv2D)	(None, 32, 32, 32)	9,248
dropout (Dropout)	(None, 32, 32, 32)	0
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_10 (Conv2D)	(None, 16, 16, 64)	18,496
conv2d_11 (Conv2D)	(None, 16, 16, 64)	36,928
leaky_re_lu_10 (LeakyReLU)	(None, 16, 16, 64)	0

max_pooling2d_9 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_12 (Conv2D)	(None, 8, 8, 128)	73,856
conv2d_13 (Conv2D)	(None, 8, 8, 128)	147,584
...

Total params: 1,316,356

Trainable params: 1,316,356

Non-trainable params: 0

Esta tabla genera un resumen de cada capa con su respectivo tipo, forma final y el numero de parámetros involucrados.

Las celdas finales indica el total de parámetros que están en la red, y distingue entre los que son entrenables y no entrenables.

Extra

El modelo puede recibir multiples parámetros para hacerlo aun mas preciso, sin embargo se debe conocer que esto requiere mayor fuerza de equipo de computo.

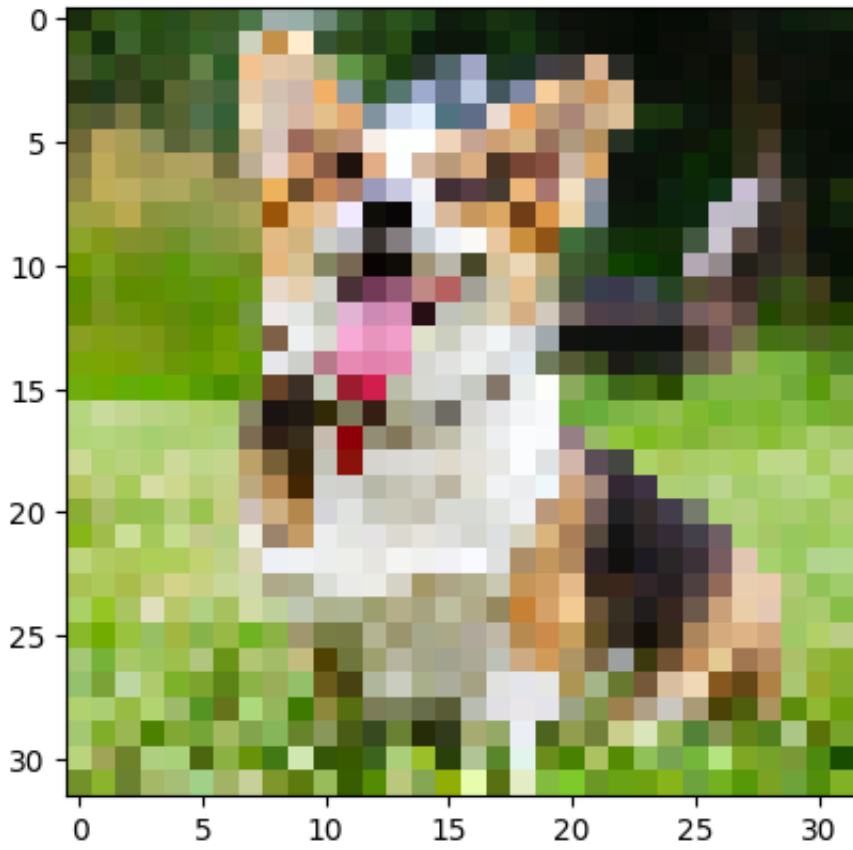
Precision vs Perdida

Loss Function (Test Loss), indica que tan bien el modelo predice la data que no ah visto durante el entrenamiento.

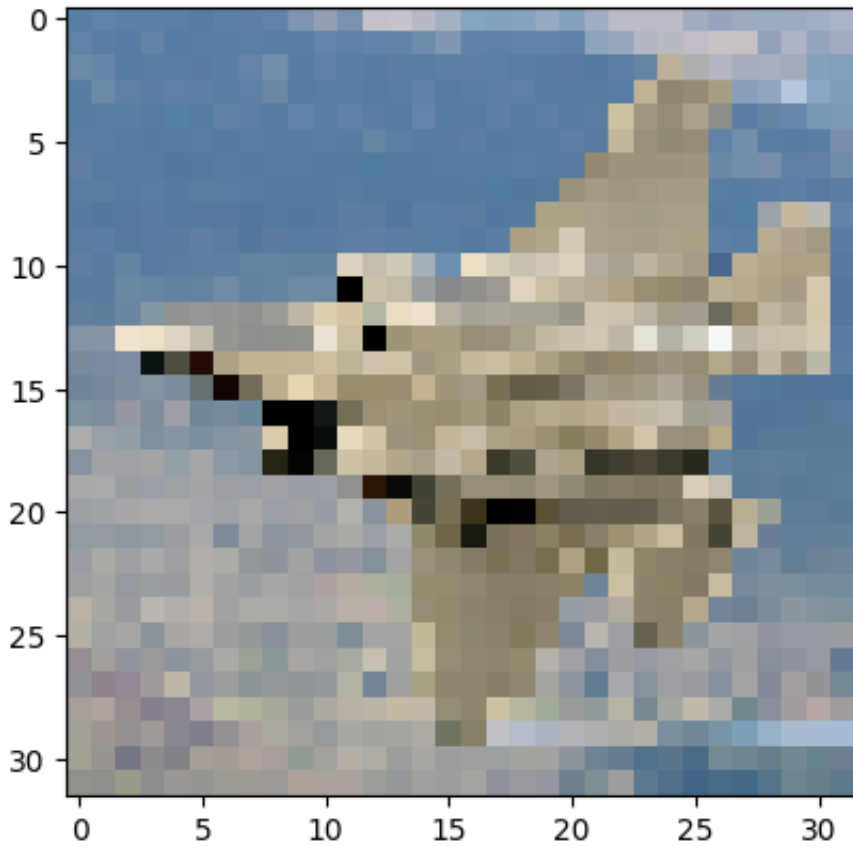
Test Accuracy, representa el porcentaje de predicciones correctas hechas por el modelo para la data que no ah visto.

```
test_loss, test_acc = model_CNN.evaluate(X_test_scl, onehot_test, verbose=False)
print("Test loss:", test_loss)
print("Test accuracy:", test_acc)
```

Test loss: 5.0816216468811035 **Test accuracy:** 0.41100001335144043



Predicted class: spider <matplotlib.image.AxesImage at 0x7fde2207c0a0>



1/1 [=====] - 0s 361ms/step Predicted class: dinosaur
 <matplotlib.image.AxesImage at 0x7fde221618b0>

Reevaluacion

```
from keras.losses import categorical_crossentropy
```

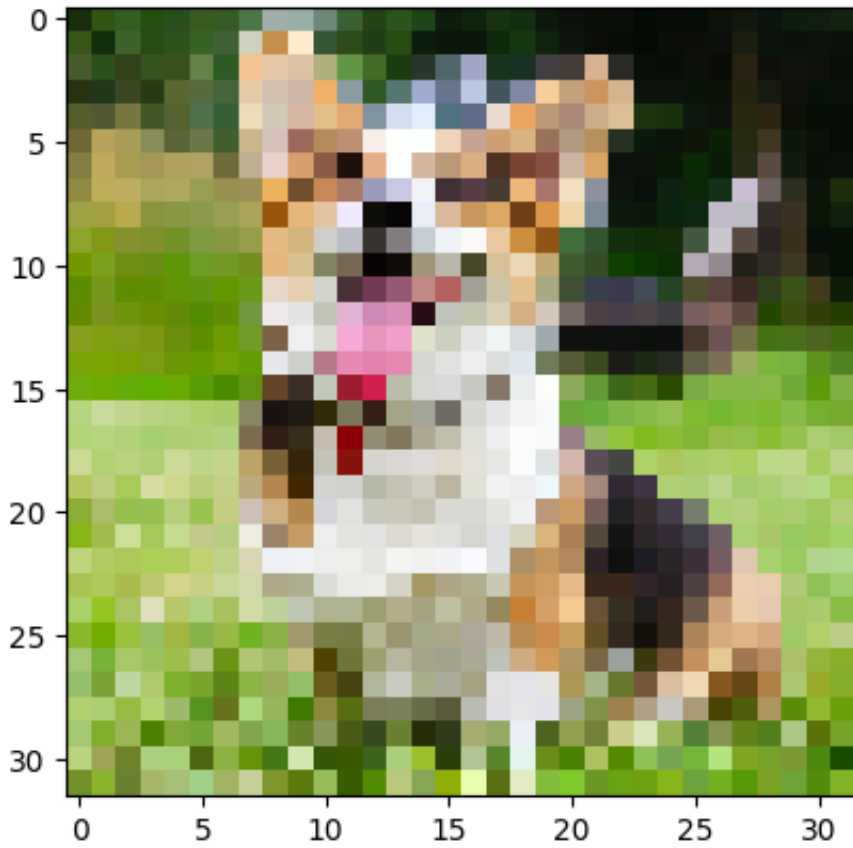
```
model_CNN.compile(loss= 'categorical_crossentropy', optimizer=Adam(), metrics=['a  

model_histo = model_CNN.fit(X_train_scl, onehot_train, batch_size=batch_size, epo
```

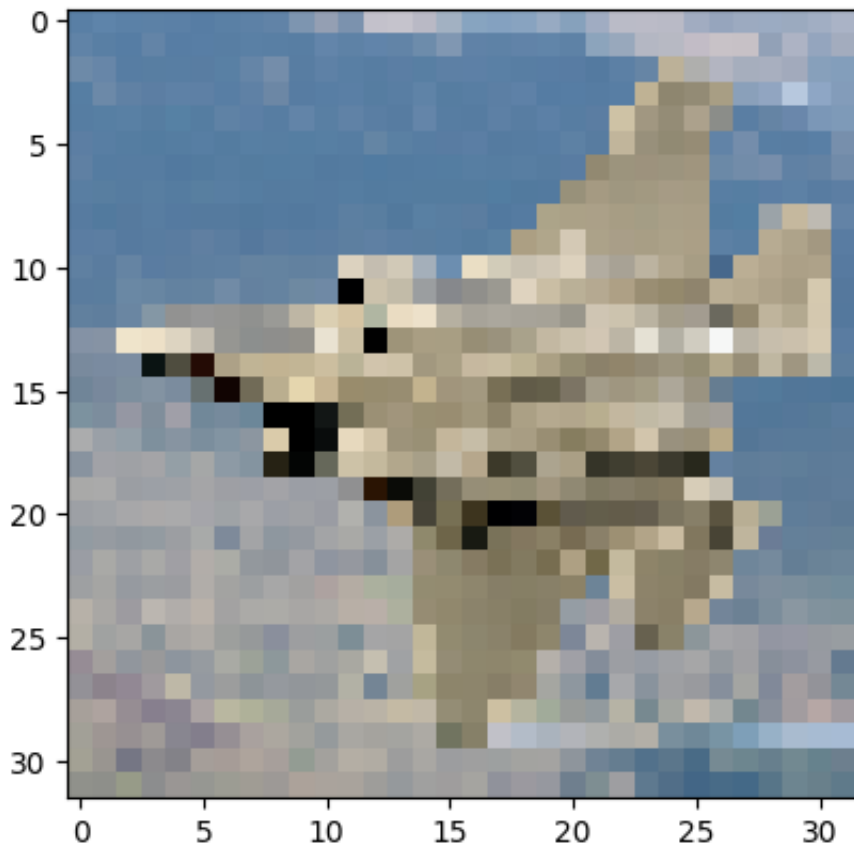
Las predicciones fueron erróneas por lo cual se procede a realizar un entrenamiento mas intensivo utilizando **Convolutional Neural Network**

Resultados de la reevaluacion

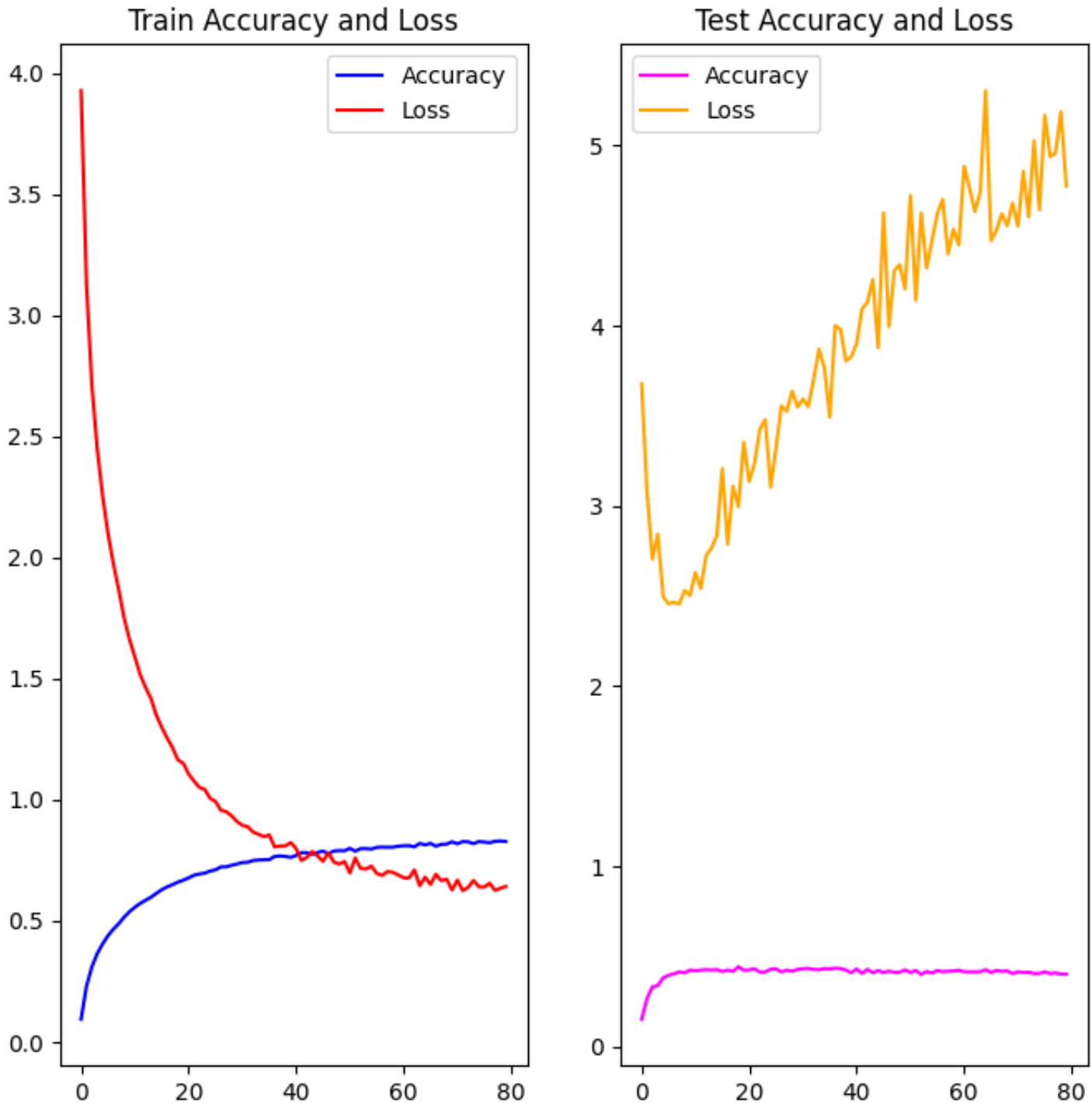
Test loss: 5.0816216468811035 **Test accuracy:** 0.41100001335144043



Predicted class: bee <matplotlib.image.AxesImage at 0x7fde2207c0a0>



1/1 [=====] - 0s 361ms/step Predicted class: plane
<matplotlib.image.AxesImage at 0x7fde221618b0>



Conclusiones

Al preprocesar los datos de CIFAR-100 y lograr implementar una Red Neuronal Convolutiva se puede lograr observar como mediante utilización de optimizadores, funciones de pérdida de entropía cruzada y entrenamiento efectivo. El proceso analítico y de evaluación se volvía cada vez más consistente para lograr soluciones de aprendizaje automático de alto rendimiento.

Fashion MNIST

Introduccion

Utilizando como base el **dataset** de **Fashion MNIST**, se hará clasificacion de los procesos y resultados, enfocados en la clasificación de imágenes.

Objetivo

- Desarrollar una red neuronal capaz de identificar y categorizar diferentes prendas de vestir utilizando imágenes como base de aprendizaje.

Dimensiones

Shape of train_images: (50000, 28, 28) Shape of train_labels: (50000,)

Análisis Estadístico de la Descripción de Imágenes

- Tabla utilizando `df_train_images.describe()`

	0	1	2	3	
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	0.000900	0.006160	0.030940	0.107540	0.253580
std	0.100893	0.269673	0.800147	2.558106	4.300201
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	16.000000	36.000000	119.000000	164.000000	224.000000

8 rows × 784 columns

La tabla generada utilizando `df_train_images.describe()` brinda un resumen estadístico para el dataset de los valores de pixels de imágenes, brindando valores provisionales como lo son `moda` , `desviación standard std` , `rango mínimo y Maximo` , y nos brinda la capacidad de conocer el nivel de `dispersion` y `tendencia` de los datos `centrales` .

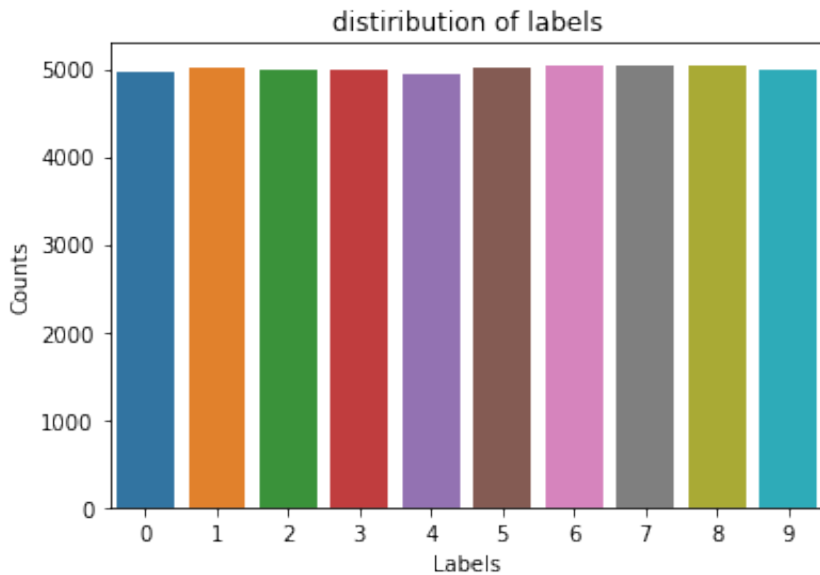
Valores Perdidos

0

Los valores perdidos son datos que al momento de `entrenar/analizar` son `omitidos` o no pueden ser `analizados`, el objetivo principal es llegar lo mas cercano a 0 .

Etiquetadores

Indice	Valor
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



Esta distribución indica que prendas se asocian con los etiquetadores otorgados.

Creacion de modelos y métricas

```
def yeniModelOlustur(input_shape=img_shape):  
    model = Sequential([  
        base_layer(16,input_shape),  
        Conv2D(16,3,padding='same',activation='relu'),  
        conv_layer(32),  
        Dropout(0.1),  
        conv_layer(64),  
        Dropout(0.2),  
        conv_layer(128),  
        Dropout(0.25),  
        conv_layer(256),  
        Conv2D(256,3,padding='same',activation='relu'),  
        Dropout(0.3),  
        MaxPool2D(pool_size=(2,2),padding='same'),  
        Flatten(),  
        dense_layer(128,0.6),  
        dense_layer(64,0.4),  
        output_layer(10)  
    ])  
    return model
```

```
m = yeniModelOlustur()
m.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
m.summary()
```

Resultado

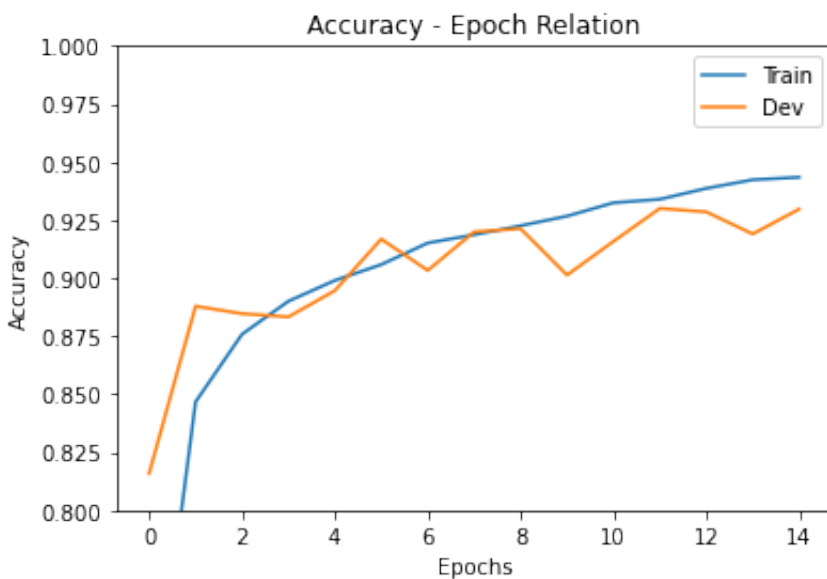
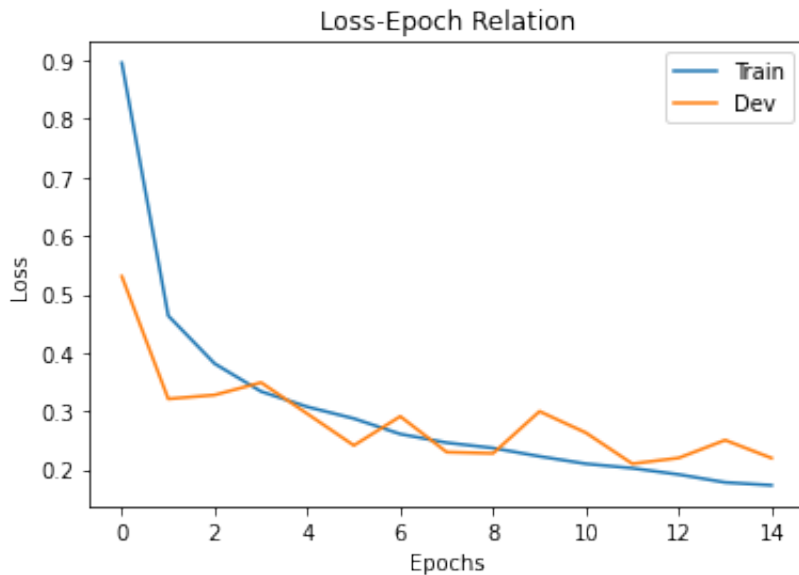
Layer (type)	Output Shape	Param #
InitLayer (Sequential)	(None, 28, 28, 16)	160
conv2d_1 (Conv2D)	(None, 28, 28, 16)	2320
sequential (Sequential)	(None, 14, 14, 32)	14016
dropout (Dropout)	(None, 14, 14, 32)	0
sequential_1 (Sequential)	(None, 7, 7, 64)	55680
dropout_1 (Dropout)	(None, 7, 7, 64)	0
sequential_2 (Sequential)	(None, 3, 3, 128)	221952
dropout_2 (Dropout)	(None, 3, 3, 128)	0
sequential_3 (Sequential)	(None, 1, 1, 256)	886272
conv2d_10 (Conv2D)	(None, 1, 1, 256)	590080
dropout_3 (Dropout)	(None, 1, 1, 256)	0
...
Total params: 1,813,050		
Trainable params: 1,811,706		
Non-trainable params: 1,344		

Esta tabla genera un resumen de cada capa con su respectivo tipo, forma final y el numero de parámetros involucrados.

Las celdas finales indica el total de parámetros que están en la red, y distingue entre los que son

entrenables y no entrenables.

Evaluación de Pérdidas y Precisión



- En la primera grafica se visualiza la perdida del modelo en el entrenamiento y validación, mientras prosigue con los temas sets de entrenamiento.
- En la segunda gráfica se visualiza la precision del modelo para ambos sets de datos, dando indicios del la eficacia de aprendizaje con el modelo.

Fine Tuning

1. Early Stopping

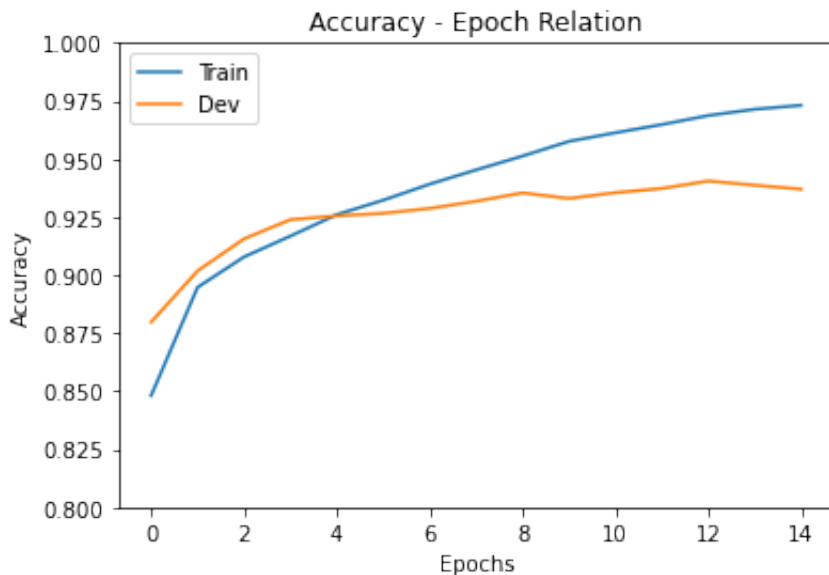
```
checkpoint_fm = ModelCheckpoint("fashion_mnist_model.h5", save_best_only=True)
early_stopping_fm = EarlyStopping(patience=10, restore_best_weights=True)
```

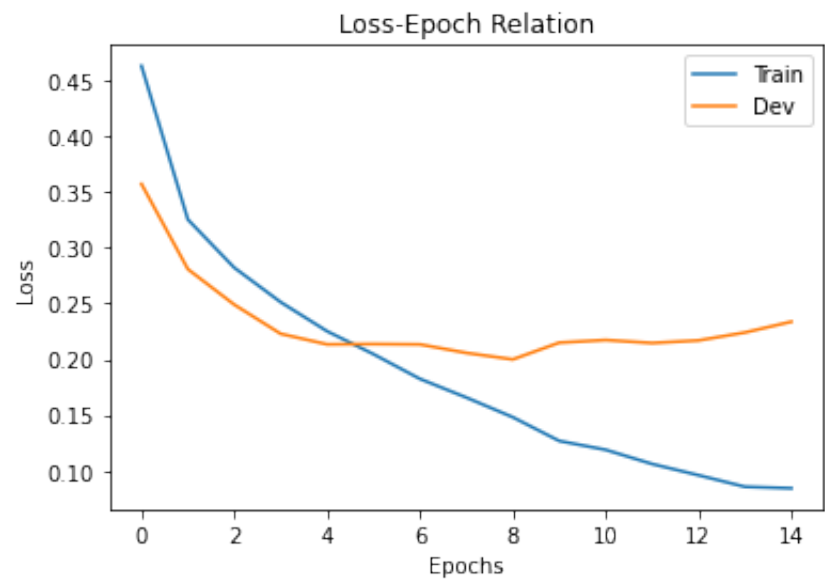
2. Learning Rate Decay

```
def exponential_decay(learning_rate, decay_step):
    def exponential_decay_fm(epoch):
        return learning_rate * 0.1 ** (epoch / decay_step)
    return exponential_decay_fm
```

```
exponential_decay_fm = exponential_decay(0.01, 10)
lr_scheduler = LearningRateScheduler(exponential_decay_fm)
```

Re-evaluacion después de aplicar FINE-TUNING





Resultados Finales

Dataset	Accuracy
Train	0.9730799794197083
Dev	0.9369937181472778
Test	0.9352999925613403

Conclusiones

Se puede concluir que el modelo ah sido desarrollado, entrenado y optimizado metódicamente para clasificar imágenes usando el **dataset Fashin MNIST**.

Los resultados iniciales indicaron un rendimiento prometedor, el cual fue validado intensivamente.

Las métricas de visualización de perdida y precision sobre los segmentos de entrenamiento proveen demostraciones de la aplicación de los principios de **DeepLearning** y de los protocolos de evaluación.